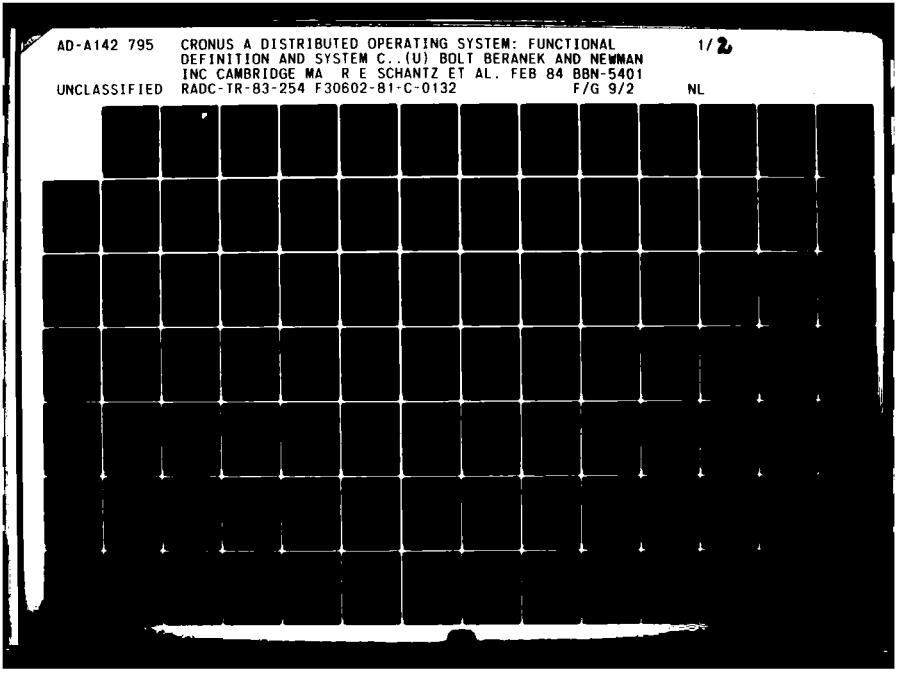
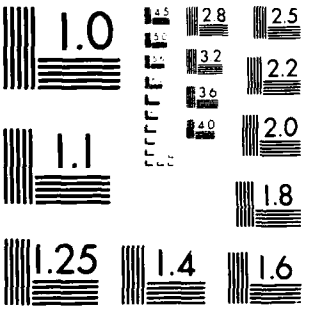

AD-A142 795 CRONUS A DISTRIBUTED OPERATING SYSTEM: FUNCTIONAL DEFINITION AND SYSTEM C..(U) BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA R E SCHANTZ ET AL. FEB 84 BBN-5401
UNCLASSIFIED RADC-TR-83-254 F30602-81-C-0132 F/G 9/2 NL
1/2



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS [illegible]

(13)

AD-A142 795

RADC-TR-83-254
Phase Report
February 1984

CRONUS, A DISTRIBUTED OPERATING SYSTEM: FUNCTIONAL DEFINITION AND SYSTEM CONCEPT

Bolt Beranek and Newman, Inc.

Richard E. Schantz, Morton D. Hoffman, William I. MacGregor
and Robert H. Thomas

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC
ELECTE
JUL 1 1 1984
S D
E

DTIC FILE COPY

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

84 07 05 091

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-83-254	2. GOVT ACCESSION NO. AD-A142795
4. TITLE (and Subtitle) CRONUS, A DISTRIBUTED OPERATING SYSTEM: FUNCTIONAL DEFINITION AND SYSTEM CONCEPT	5. TYPE OF REPORT & PERIOD COVERED Phase Report Jun 81 - Jun 82
	6. PERFORMING ORG. REPORT NUMBER Report No. 5401
7. AUTHOR(s) Richard E. Schantz William I. MacGregor Morton D. Hoffman Robert H. Thomas	8. CONTRACT OR GRANT NUMBER(s) F30602-81-C-0132
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman, Inc. 10 Moulton Street Cambridge MA 02238	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 63728F 25300107
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (COTD) Griffiss AFB NY 13441	12. REPORT DATE February 1984
	13. NUMBER OF PAGES 124
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Thomas F. Lawrence (COTD)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Distributed Operating System
Local Area Network
Distributed System

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report details the functional definition and system concepts for Cronus, the local area network distributed operating system being developed as part of the DOS Design and Implementation project sponsored by Rome Air Development Center. The report is the first project deliverable document, and is intended as an overview of the system which we will be developing under this effort. The functions and system concepts discussed in this report are the results of a consideration of

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

the current state of distributed system technology and potential uses of the system in a wide variety of command and control environments.

Table of Contents

1 Introduction 1
1.1 Project Objectives 2
1.2 System Environment 7
1.3 System Goals 10

2 Coherence and Uniformity 13
2.1 The Outer System and Inner System Views 13
2.2 DOS Cluster Physical Model 19
2.3 Design Principles 21
2.3.1 Provide Essential Services System-Wide 21
2.3.2 Utilize Recognized and Emerging Standards 22
2.3.3 Preserve Choices 23
2.4 Specific Approaches 24
2.4.1 The Communication Subsystem 24
2.4.2 Generic Computing Elements 25
2.4.3 Standards Applicable to DOS Components 27
2.4.4 Flexible Application Host Integration 28
2.4.5 Comprehensive DOS Object Model 29
2.5 A Summary of the DOS Architecture 31
2.5.1 Level 1: The Minimal System 32
2.5.2 Level 2: The Utility System 32
2.5.3 Level 3: The Application System 33

3 The DOS Functions and Underlying Concepts 37
3.1 Introduction 37
3.2 System Access 40
3.3 Object Managememt 43
3.4 Process Management 45
3.5 Authentication, Access Control, and Security 46
3.6 Symbolic Naming 52
3.7 Interprocess Communication 56
3.8 User Interface 58
3.9 Input / Output 61
3.10 System Monitoring and Control 63

4 System Integrity and Survivability 65
4.1 Reliability Objectives 68
4.2 General Approach 68
4.3 Specific Approach 70

5 Scalability ... 75
5.1 General Approach ... 75
5.2 Specific Approach ... 77

6 Global Resource Management ... 81
6.1 Objective ... 81
6.2 General Approach ... 82
6.3 Specific Approach ... 83

7 Substitutability of System Components ... 87
7.1 Objective ... 87
7.2 Approach: Use of Abstract Interfaces ... 88
7.3 Approach. Specific Interface Plans ... 89

8 Operation and Maintenance ... 95

9 Test and Evaluation ... 97
9.1 Coherence and Uniformity ... 98
9.2 Integrity and Survivability ... 101
9.3 System Scalability ... 102

10 Relation to OSI TAFIIS Report ... 105
10.1 General Aspects of the OSI Model ... 105
10.2 OSI Identified Functions ... 107

11 DOS Glossary ... 111

FIGURES

The Generic Computing Element 26
The Local Cluster Configuration 34
The InterCluster Environment 35
The DOS Security Envelope 50

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1
DTIC COPY INSPECTED 2

1 Introduction

This report details the functional definition and system concepts for Cronus, the local area network distributed operating system being developed as part of the DOS Design and Implementation project sponsored by Rome Air Development Center. The report is the first project deliverable document, and is intended as an overview of the system which we will be developing under this effort. The functions and system concepts discussed in this report are the results of a consideration of the current state of distributed system technology and potential uses of the system in a wide variety of command and control environments.

This report is not a design document. The design of a system meeting the objectives described in this report will be covered in later reports. However, the nature of the project dictates that many design, implementation and even test and evaluation approaches be made in a coordinated manner with the system definition. Accordingly, these issues are also addressed where appropriate in the present document.

1.1 Project Objectives

The purpose of the Distributed Operating System (DOS) project is to develop a distributed operating system for use in command and control environments. The DOS development activity can be subdivided into five major categories:

1. Select the off-the-shelf hardware and software components that represent the foundation of the DOS system.
2. Design the DOS conceptual structure, by defining a) the functions available to users of the system, b) models for pervasive issues such as reliability, security, and system control, and c) the top-level decomposition of the DOS software components into implementation units.
3. Develop the implementation units defined in (2), until they become complete, functioning programs in the DOS Advanced Development Model (ADM).
4. Integrate the implementation units into a coherent and useful system, both by adjustments to the functional definitions and by any optimizations necessary for acceptable performance.
5. Evaluate the concepts and realization of the DOS in the environment of the ADM, by means of formalized test procedures and practical demonstrations.

The DOS will be designed as a general purpose system to support interactive information processing. Thus, emphasis is placed on adaptability of the DOS structures along several dimensions, for example:

- Reliability: essential services can be provided with high reliability using redundant equipment, or with lower reliability at lower cost.
- Accommodation: there are well-defined paths for

integrating any host under any operating system, and any special-purpose device, into the DOS.

- Scalability. a DOS cluster can be scaled from a few to several hundred hosts, and adjust to a similar scaling of the user population.

- Primary use. appropriately configured, a cluster may be efficiently utilized as a program development system, an office automation system, a base for dedicated applications, or a mixture of all three.

- Access paths. the DOS services and applications can be accessed from terminals and workstations attached to a cluster directly, or through the internetwork.

- Buy-in cost. hosts and applications can be integrated into the DOS environment in a variety of ways, that offer a range of cost/performance points to the integrator.

The DOS concepts and the software modules that implement the basic system services can be utilized in a wide variety of possible hardware configurations, and in many different operating regimes, to support the requirements of different applications This polymorphous aspect of the DOS makes it difficult to describe concisely, a complete description must examine each of the dimensions of DOS adaptability. This document presents a top-level view of the project objectives and DOS design goals. further detail will be provided in system design documents

With regard to DOS adaptability, we distinguish between accommodation, the ability of the DOS to incorporate new host types, new constituent operating systems, and new application subsystems (services), and substitution, the replacement of a hardware or software module critical to the provision of DOS

essential services. It is a project goal to achieve the widest possible range of accommodation, i.e., to be able to integrate many types of existing or future host, operating system, or application subsystems within the DOS concepts. Substitution, in contrast, will be much more tightly constrained, because the new hardware or software module must correctly implement the external interface of the old module in order for the DOS to continue to provide essential services. Certain critical interfaces (e.g., the interface to the local network, GCE operating system support) will be carefully defined to make substitution possible. Both forms of adaptability, accommodation and substitution, are important, but we expect accommodation to occur much more frequently.

In general, the DOS design is influenced more by available and projected technology than by the specific requirements of any application, since to do otherwise would violate the general-purpose nature of the DOS.

The temper of the DOS design is pragmatic. The project aims to design, build, and evaluate a complete, useful system over a period of about 3 years. The use of the DOS as a tool for its own implementation is an important incentive to the developers to be on time and down-to-earth.

The following problem areas are not considered to be

important project objectives.

1. Development of high reliability or fault tolerant hardware.
2. Development of minimal-cost solutions to distributed processing problems.
3. Research into low-level communications hardware and protocols.
4. Development of support for distributed, real-time applications.

By stating (1) as a non-goal we emphasize the project orientation towards software, rather than hardware, reliability techniques. We note the mention of specific, non-fault-tolerant commercial processors as DOS constituent hosts in the Statement of Work, the implication that non-fault-tolerant machines will often be included in DOS configurations is evidence in support of (1) as a non-goal.

By stating (2) as a non-goal we express a bias towards general-purpose operating system facilities. For some applications, high-volume production (hundreds or thousands of units) may be anticipated, economic pressures will then encourage tailoring the systems to provide the required function at minimal cost per unit. General-purpose systems, on the other hand, tend to provide more functionality than is necessary for any particular application. They are thus more cost effective for small production volumes of application systems (their generality

makes programming less costly), and less cost effective for large production volumes (since each replicated system contains unused general-purpose mechanisms). Because simply achieving the required distributed operating system function is to a large degree still a research problem, we do not believe a major emphasis on cost effectiveness is desirable or even possible at this time.

By stating (3) as a non-goal we recognize the large investments in low-level communication protocols and hardware already made by the Department of Defense and the private sector. In the interests of interoperability and a rapid rate of progress on the other, higher-level issues of distributed operating system design, we will directly utilize the *DoD IP and TCP protocol* standards and commercial local network technology.

By stating (4) as a non-goal we identify a conflict between the distributed operating system structures required for high performance in real-time systems and the structures which support a modern, general-purpose computing utility. Again, the project orientation is towards the more general-purpose concepts; however, the presence of individual hosts in a DOS cluster performing real-time processing is entirely within the DOS concept of operation, and is readily supported.

1.2 System Environment

To define the focus of the DOS project it is useful to classify distributed systems along architectural lines according to the physical extent of distribution the systems exhibit We can identify three major architectural areas of interest

1. Node Architecture
2. Cluster Architecture <1>
3. Inter-Cluster Architecture

Each of these is related to the emerging technology of distributed systems, but the technology of distribution tends to be different in the three areas, as explained below

Node Architecture

The development of a processor architecture, configuration, and operating system for a single host or processing node is a large-scale undertaking, usually accomplished by computer manufacturers. A host is typically physically small (can be contained in one room), is designed by computer hardware architects as a single logical unit, and is concerned with maximum event rates of approximately 1 to 1000 million events per second. Although dual-processor nodes have been common for some time, nodes with many-fold internal distribution are just now becoming commercially available The structure and efficient utilization of such hosts is at

<1>. The term "cluster" is used here with the same meaning as in BBN Report No 4455, "Distributed Operating System Design Study Phase 1". The term "cluster architecture" is synonymous with the term "MINIDOS" in the OSI Report No. R79-045, "TAC C3 Distributed Operating System Study Final Report", similarly, "inter-cluster architecture" is synonymous with "MAXIDOS" in the OSI report

the forefront of computer architecture research

Cluster Architecture

A cluster is a collection of nodes attached to a high-speed local network. At present, technology limits the speed of local networks to approximately 1 to 100 megabits aggregate throughput, and the physical size of the network to a maximum diameter of about 4 kilometers. The host systems are made to work together through the agency of the distributed operating system, which provides unifying services and concepts which are utilized by application software. The maximum event rate at the DOS level is related to the minimum message transmission time between hosts, and is on the order of 10 to 1000 messages per second. The cluster configuration and applications supported by it are typically under the administrative control of one organizational entity.

Inter-Cluster Architecture

An inter-cluster architecture typically deals with geographically distributed clusters (or in the degenerate case, hosts) which are not under unified administrative control. Because of administrative issues and the greater lifespan of inter-cluster architectures, they tend to be composed of parts from many different hardware and software technologies. The communication paths between widely separated clusters have much lower bandwidth and higher error rates than local networks; the maximum event rate for cluster-to-cluster interactions is on the order of 0.01 to 10 events per second. In the inter-cluster case, emphasis is on defining protocols for interactions between clusters, and on the appropriate rules for the exchange of authority (for access to information and consumption of resources) between clusters.

The boundaries between these areas are often indistinct, and sometimes simply the result of unrelated design efforts. Nonetheless each area has a unique set of requirements and

solutions for system design. For a given area, these aspects combine to form an outlook that encompasses not just the functional properties of a system, but also many "system level" issues relating to development, administration, training, operations, documentation, and maintenance.

The principle concern for the DOS project will be the development of a system for a cluster architecture. Because a cluster is composed of nodes and connected to other clusters the relationships between node, cluster, and inter-cluster architectures must be considered in order to produce the DOS cluster architecture. In certain specific but limited regards, problems concerning node or inter-cluster architecture will be important. For example, it must be possible to integrate a wide variety of nodes into the cluster system, and the cluster system must be able to interact with other clusters. Where feasible, the design will accommodate existing standards in the areas of node and inter-cluster architecture. Standardized node components and standardized connections to the internetwork environment will both contribute to the applicability and longevity of the DOS design. However, it is outside the scope of this project to attempt the development of unified approaches to problems of distribution in all three areas, which would involve addressing three different sets of issues.

It is important that the DOS project take full advantage of the best available off-the-shelf component technology A "component" in this sense may be hardware (e.g. processors and storage devices) or software (e.g., the commercial UNIX or VMS operating systems, and the ARPA-sponsored internet gateway software) The current technological trends should also favor continued development of the components in applications apart from the DOS project, so that the parallel evolution of node and inter-cluster architectures can potentially benefit the DOS cluster architecture The DOS project can be expected, of course, to provide useful concepts and services for the other areas, synergism results from a blend of diversity and commonality among the three architectural levels

1.3 System Goals

The overall objective of developing a cluster operating system can be broken down into a number of system design goals along the lines of the characteristics the system should exhibit The resulting design goals can then be prioritized and used during the design process as a means for focusing the design effort and as a basis for making various design choices

The system design goals for the DOS, in order of decreasing

priority are

Primary Goals

1. Coherence and Uniformity.

   To be usable as a system the DOS must implement a coherent and uniform user model.

2. Survivability and Integrity.

   The operation of the system and the integrity of the data it manages must be resilient to outages of system components.

3. Scalability.

   It should be possible to configure the system with varying amounts of equipment to accommodate a wide range of user population sizes and application requirements.

Secondary Goals

4 Resource Management.

   The system should provide means for system administrators to establish policies that govern how resources are allocated to user tasks, and it should work to enforce those policies

5. Component Substitutability

   The ADM DOS will be built on a specific equipment base The system should be structured to permit system components to be replaced by functionally equivalent equipment to the largest extent feasible

6 Operation and Maintenance Procedures

   The system should provide features which facilitate routine operations and maintenance activity by system operations personnel

Each of these design goals is discussed in more detail in the sections that follow.

The distinction between primary and secondary goals is methodological, and principally related to test and evaluation. Each primary goal will be the subject of a well-defined evaluation procedure, specified to the greatest extent practical in advance of system implementation. For example, the tests for survivability will include a prescribed set of failure modes to be artificially induced in the Advanced Development Model, and the behavior of the system recorded. The success of the DOS design in meeting secondary goals will not be so carefully scrutinized, written evaluations will be prepared, but less effort will be spent on planning the evaluations and producing comprehensive tests.

2 Coherence and Uniformity

The DOS project aims to develop a coherence and uniformity among otherwise independent application systems and computer services attached to a cluster, in such a manner that the effort required to develop composite applications from existing applications, or to develop new, integrated applications, is as small as practical.

This section discusses "coherence and uniformity" as the phrase applies to the DOS. First, an important dichotomy in the domain of anticipated DOS applications is explained, and the tensions that this dichotomy places on the design process are described. Second, the cluster architecture is described in more detail. Third, several design principles which are the basis of the design process are presented and discussed as they apply to the goal of coherence and uniformity. Finally, specific approaches to some of the issues which are believed to be well understood at the current time are given.

2.1 The Outer System and Inner System Views

The interpretation of the phrase "coherence and uniformity" is ultimately subjective, and should reflect the end-users opinions of the system concepts and realization. Thus it is

fitting that this section begin with a discussion of how the DOS concepts might be used in different applications. Rather than attempt a thorough treatment of the (very large) domain of applications, two important classes of applications are considered in the abstract.

The first class of application views the DOS as an external entity, a supplier of services and communication facilities. This orientation is referred to as the outer system view of the DOS, since the applications already exist or are built outside the context of the DOS concepts of operation. The second class of application is built to run in the DOS context, with full knowledge of the DOS environment and a bias towards its full exploitation. This orientation is referred to as the inner system view of the DOS. The outer system view is most closely related to the problem of achieving connections among existing functional components built on heterogeneous hosts and operating systems; the inner system view should prevail in the design of new, distributed applications, whether they are built on a homogeneous or heterogeneous base.

We presume that applications satisfying an organization's needs will often be developed independently of each other. During their development, these applications will frequently come to depend upon particular hardware and software objects in their

environment, e g , the host instruction set, the host operating system, and one-of-a-kind peripherals attached to a particular host The applications may reach operational status with no explicit use of the DOS concepts, and they could be built either on conventional, stand-alone hosts or on a host attached to a DOS cluster

At some point in time it may be necessary to form a logical connection between application programs which have been developed independently--that is, to achieve interoperability among the functional components There may be many obstacles to interoperability, a few of the more prevalent and difficult obstacles are

1. Incompatible data structures.
2. Application interfaces designed for program-to-human rather than program-to-program communication
3. The absence of a suitable program-to-program communication facility in the host operating system(s)
4. An inadequate structure for the transfer of authority (for access to information and resources) between programs.
5. Poor reliability as the system becomes more and more vulnerable to single-point failures.
6. Poor reliability due to high error rates on communication channels between components.
7. The high cost of performance optimizations involving several complex software components.
8. Disparate software development environments--both automated tools and manual procedures

In the outer system view, the primary role of the DOS is to reduce these and other obstacles to interoperability, by providing a core of common concepts and functions that become the focus of component interactions.

As an example of the outer system view, suppose there is a need to link a graphics display function executing on a personal workstation to a database management system running on a standard mainframe operating system. Initially, the database management system and the graphics support may have no relationship to the DOS whatsoever, relying entirely upon the hardware and software resources of their own hosts. In order to accomplish the logical link, the hosts must be physically attached to a DOS cluster, communication software must exist on each host, and the applications must be prepared to properly utilize the host-to-host communication path. The DOS can assist this integration by defining the common concepts required for the logical connection to be formed. In this simple example the only requirement is for communication, but in more complex situations the DOS may supply other services (e.g., user authentication, data storage and encryption, terminal multiplexing).

The inner system view, in contrast, assumes that new applications are constructed within the framework of the DOS and use DOS mechanisms in preference to local host mechanisms.

whenever practical. A new application designed from the inner system perspective may or may not be distributed, and may be built on homogeneous or heterogeneous machines and operating systems. Whichever the case, by adopting the DOS conventions for process control, file storage and cataloging, and process-to-process communication (among others) such applications avoid many of the interoperability problems listed above. In fact, the process of building an application on the DOS inner system is akin to program construction on a single conventional host, in that the concepts of "process" and "file" and "directory", to name a few, are generally understood by all of the components to mean the same thing. The new application not only achieves uniform connections among its constituent pieces, but also inherits the ability to interact with other inner system tools which also conform to the common concepts. Thus inner system applications enrich the DOS environment in an incremental way, and form the basis for the continued orderly evolution of the DOS environment.

The DOS inner system is unlike a conventional operating system, however, because it addresses issues of distribution--the development of distributed programs, the possibility of survivable operation through host redundancy, and the potential for configuration scaling beyond the limits of shared memory architectures. These system aspects motivate the development of

a powerful and coherent inner system architecture

Brief examples will reinforce the distinction

An example of an outer system view might involve two components a commercial DBMS running on a standard mainframe operating system, and a workstation generating color graphics displays The objective of the application is to provide a facility for online, color graphics displays of data stored in the DBMS This is an outer system problem, because the DBMS and mainframe operating system (and quite possibly the workstation operating system) are large and complex objects, maintained by independent organizations, difficult to modify, and were constructed with no awareness of the DOS The most conservative (least implementation effort) approach might use the DOS only as a communication path, and achieve only minimal integration of the mainframe host into the DOS

An example of the inner system view might involve the construction of a programming environment for a new microprocessor The DOS already contains many program development tools--editors, compiler-compilers, linkers, debuggers, etc By adopting the DOS concepts for process interaction, many or most of these may be reused <1>

<1> The UNIX operating system is widely regarded as a good example of the inner system view, shell programming, the "makefile" facility, and other system facilities contribute to

A fundamental assumption of the DOS project is that both the outer and inner system views are important and must be considered in the design. Because the two views imply different system requirements this represents a burden to the design process.

2.2 DOS Cluster Physical Model

Before discussing the major system design principles, the equipment configuration for the DOS cluster is briefly reviewed.

The DOS cluster is composed of three types of equipment:

1. A communication subsystem. The subsystem consists of a high-bandwidth, low-latency local network, hardware interfaces between hosts and the local network, device driver software in the host operating systems, and low-level protocol software (the data link layer) in the hosts.

2. DOS service hosts. These machines are dedicated entirely to DOS functions, and exist only to provide services to DOS users and applications. In general, they represent modules with specific, system-oriented functions (e.g., archival file storage) and are not user programmable. Requirements for the DOS service host types and operating systems will be specified in the DOS design documents.[1]

---

the growth of UNIX systems by accretion.

[1] The DOS design will permit the substitution of different service host types for the hosts actually used in the Advanced Development Model, however, any substitution must meet minimal requirements specified in the concept of operation.

3 Application hosts These may be general-purpose hosts (e g , timesharing machines) providing services to many DOS users, or workstations providing services to one user at a time, or special-purpose hosts (e g signal processing computers) required by just one DOS application Application hosts are often user programmable In general, they have many characteristics which are not under the control of the DOS the DOS must be sufficiently flexible to incorporate application hosts of almost any kind

Application hosts will be connected to the communication subsystem in one of two ways 1) directly, by means of a host-to-local-network device interface, or 2) indirectly through an intermediary DOS service host called an access machine The intent is that standardized access machine software and hardware can reduce the integration cost for a new application host The electrical interface between the application host and the access machine, for instance, need not be as complex as a local network interface, it need only be mutually acceptable to the two machines <1> Access machines may have other functions as well they could play a role in the DOS security model, for example by isolating untrusted hosts from the (presumed secure) local network. The tradeoffs arising in direct and indirect host integration are not presently well understood, an exploration of this topic is a DOS project goal

General-purpose application hosts will usually operate with

<1> As a concrete example, the access machine planned for the Advanced Development Model will utilize the HDLC protocol over an RS-422 or RS-423 machine interface

standard operating systems (e.g., a Digital Equipment Corporation VAX computer running the VMS operating system) which are enhanced and/or modified to integrate the host into the DOS. Thus application hosts will support some DOS software components, at a minimum those required for communication with DOS service hosts. Some DOS services may also be partially or completely implemented on application host to realize performance advantages (by locating applications and required DOS services together) or cost advantages (through resource sharing).

2.3 Design Principles

2.3.1 Provide Essential Services System-Wide

At the heart of the DOS concept is the availability of selected, essential services to all of the applications in the DOS. The coherence and uniformity of the DOS is directly enhanced when applications and application host operating systems embrace the DOS-supplied services as the single source of these services. To the extent that applications and application host operating systems choose to utilize parallel but incompatible services, coherence and uniformity is lost.

At this time the essential services are believed to be

- User access points (terminal ports, workstations) providing a uniform path to all DOS services and applications
- Object management (cataloging and object manipulation) for many types of DOS objects.
- Uniform facilities for process invocation, control, and interprocess communication for application builders.
- Cluster-wide user identifiers and user authentication as the basis for uniform access control to DOS resources.
- Cluster-wide symbolic name space for all types of DOS objects.
- A standard interprocess communication (IPC) facility supporting datagrams and virtual circuits.
- A well-designed user interface that provides access to all DOS and application services.
- Input/output services for the exchange of data with people and systems apart from the DOS.
- Host monitoring and control services, and additional mechanisms needed for cluster operation

2.3.2 Utilize Recognized and Emerging Standards

The DOS design will incorporate recognized and emerging standards whenever practical at many levels of the system. The adoption of standards both enhances the uniformity of the system and contributes to the likelihood of pre-existing, compatible interfaces. The longevity of the DOS concept of operation is extended by attention to standards that are the foundation of contemporary research and development activities, the possibility of interaction with other projects to the mutual benefit of both

is maximized.

2.3.3 Preserve Choices

The DOS design will preserve choices for the application host integrator and the application builder.

There is a complex tradeoff between the cost of host and application integration into the DOS, and the uniformity and power achieved as a result of the integration. Although many issues involved in the tradeoff have been identified, the problem is not sufficiently well understood to make prescriptions confidently. Investigation of this problem is an important objective of the DOS project.

Part of the project's approach is embodied in Principle 3. This principle requires that the DOS concept of operation accommodate not just one, but a range of possible cost uniformity points.

Similar tradeoffs exist among the DOS services supplied to application programs. For example, this principle applied to interprocess communication implies that neither datagram nor virtual circuit service is sufficient for all applications; the DOS should provide both types of communication service.

In general, this principle requires that the DOS design address the problem of how DOS installations will adapt to very different configuration and application requirements

2 4 Specific Approaches

2 4 1 The Communication Subsystem

A high-bandwidth, low-latency local network <1> is the backbone of the DOS. The DOS concept of operation will specify the interface to the local network, so that alternate local network technologies can be substituted for the particular local network chosen for the Advanced Development Model, if they meet the interface specification. The interface specification will be as unrestrictive as possible, so that substitution is a real possibility

The local network will permit every host to communicate with every other host in the DOS cluster, and will provide an efficient broadcast service from any host to all hosts. The local network interface specification may further restrict the minimum packet size, addressing mechanism, and other local network properties

---

<1> See DOS-Note 21, "DOS Local Network Selection"

2 4 2 Generic Computing Elements

The concept of a Generic Computing Element (GCE) is important to the DOS design <1> A GCE is an inexpensive DOS host that can be flexibly configured with small or large memory and with or without disks and other peripherals, as shown in Figure 3 GCE's will be configured for, and dedicated to specific DOS service roles, such as terminal multiplexing, file storage, access machines, and DOS catalog maintenance <2>

The GCE's are the basis for implementing the essential DOS services in a uniform, application-host-independent manner Because the DOS design will specify the properties of GCE s and also the software components <3> running on them, it is possible to control the performance and reliability characteristics of the essential DOS services A configuration consisting of the local network, some number of GCE's, and supporting the essential services represents the minimum useful DOS instance

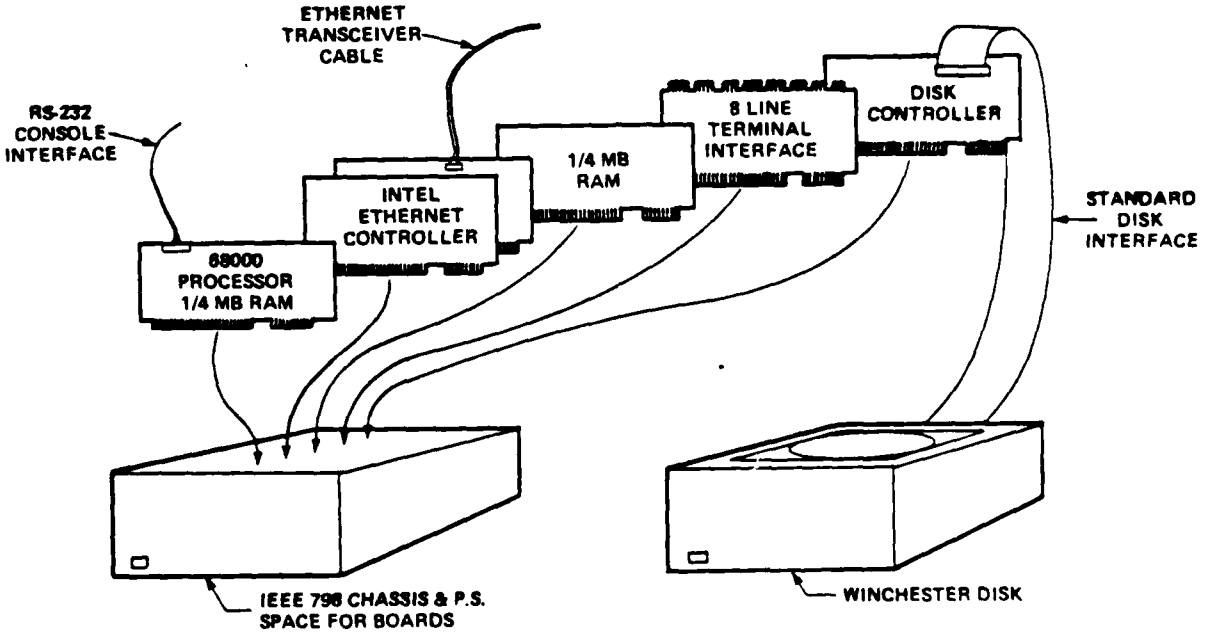
Application programs can be constructed above the GCE hardware and operating system, a single GCE host may support DOS services or user applications, but not both

---

<1> See DOS-Note 17, "A Generic Computing Element for the DOS Advanced Development Model"

<2> A single GCE may support several DOS services simultaneously

<3> Perhaps the most important software component is the GCE operating system, CMOS



The Generic Computing Element
Figure 1

2 4 3 Standards Applicable to DOS Components

The DOS design will utilize recognized standards in several key areas, these directly contribute to both the coherence of the DOS and interoperability with other computer systems The standards which have been identified as pertinent as of this time are

1. IP and TCP internet protocol standards To the maximum extent possible, IP and TCP will be used for host-to-host communication within the DOS cluster

2. ARPA standard gateway. The gateway between the ADM cluster and the ARPANET will be an LSI-11 based, ARPA standard gateway, developed and supported by BBN

3. Ethernet. From the hosts' point-of-view, the local network in the Advanced Development Model will match the Ethernet transceiver cable compatibility interface <1>

4. IEEE 796 bus. The GCE hardware selected for use in the Advanced Development Model is based on the IEEE 796 bus standard for circuit board interconnection

5. HDLC and RS-232C. These communication standards will be used to connect hosts and terminals, respectively, to GCE's within the cluster.

6. The programming language Ada The military standard language Ada will be exploited to the greatest extent practical. <2> Its use will be determined by timely completion of activities not under the control of the DOS project.

<1> As noted above, the DOS concepts will not depend upon any local network properties which are peculiar to the Ethernet Ethernet-compatible devices will, however, be easily added to the Advanced Development Model

<2>. See DOS-Note 16, "Some Thoughts on the Selection of a DOS Implementation Language"

Other standards may be applicable to DOS components and are being considered for adoption by the project. Two areas in which existing standards will probably be adopted, rather than developed by the project, are the format of electronic mail messages and the interface between GCE's and mass storage modules.

2.4.4 Flexible Application Host Integration

When a new host is integrated into a DOS cluster, it will assume one of several possible host roles. The host roles will occupy different points along the spectrum of integration cost versus degree of adherence to the DOS unifying concepts. System administrators are thus presented with a choice of integration paths, and can tailor host roles to the needs of specific applications.

When a host is integrated with minimum effort, little more than a communication path between the host and other entities in the DOS cluster will be present. This host will be able to obtain many DOS essential services through the communication path, but its resources may be unavailable to other DOS processes. Further effort must be devoted to assimilate the host partially or fully into the DOS object catalog, process model,

and reliability mechanisms

As defined above, the access machine concept is closely related to the effort required for host integration Minimal effort integration will most likely be achieved through the use of access machines This host integration path will probably result in lower throughput between the host and the network due to the presence of the access machine, but may be a desirable approach on balance For special purpose devices with limited programmability, access machines may play the dual role of device controller and DOS interface.

The host role is decided anew for each host in a cluster It is possible, for example, for two hosts which are physically the same type of machine and which run the same operating system to be integrated to assume different roles

2.4.5 Comprehensive DOS Object Model

The DOS concepts will revolve around a group of basic object types files, processes, hosts, users, and messages, to name a few of the more important The DOS design will attempt to treat all of these types (and others) uniformly, in accord with an abstract object model The abstract object model recognizes that an object may be designated by one of three varieties of name

1 Universal Identifier (UID) A UID is a fixed-length bitstring Every object in the abstract object model has a unique UID, over the set of all objects in the cluster and the entire lifetime of the system A UID is always an acceptable designator for an object within the DOS

2 Address An address is a bitstring composed of a sequence of address portions Each successive portion serves to narrow the set of objects designated by the address, the complete address refers to a single object

3 Symbolic Names People use symbolic object names to designate DOS objects Symbolic names can be context dependent (for example, relative to a directory) or context independent The symbolic name space is hierarchically structured so that the logical grouping of related objects is reflected in a similarity among their context independent symbolic names An object need not have a symbolic name.

Normally, people will refer to objects using symbolic names, and programs will refer to objects using UID's, addresses, and symbolic names The system will provide translation services the most important supported by the object catalog, to translate among the three representations of object names

UID's, addresses, and symbolic names will be used in different ways within the DOS A UID is always a sufficient object name, even for objects which can move from host to host <1>, because it is completely context independent An address will usually represent the fastest access path to an object, because its representation explicitly contains the routing

<1> The DOS does not, in general, support movement of arbitrary objects from one host to another, some specific object types will give rise to mobile objects, however

information needed to reach the designated object Symbolic names are most suitable for the user interface, but because the other object designators are available programs need not deal in general, with variable-length symbolic object names

A mechanism will be developed for constructing new, composite abstract types from previously defined types This will allow objects with rich semantics to be built from simpler objects, for example, a "reliable" file could be assembled from several primitive files on different hosts, containing redundant copies of the same information.

2.5 A Summary of the DOS Architecture

The commitment of the DOS design to support a wide range of equipment configurations makes it difficult to give a concise description of "the DOS". The system will have widely varying characteristics for different DOS equipment configurations It is possible, however, to identify three levels of 'DOS product which may help to clarify the boundaries of the design

2 5 1 Level 1 The Minimal System

The minimal DOS system consists of the local network a small number of GCE's supplying essential services, and a host integration guide which explains how the owning agency can integrate their own hosts into the DOS environment

The minimal system supports the user registration and authentication functions, and the essential services pertaining to the user interface, the object model, the cluster gateway(s) It also supports the basic system monitoring and control functions present in any DOS instance By itself, it does not provide a user programming environment, or the utilities (electronic mail, text preparation, etc) found in most timesharing environments

2 5 2 Level 2 The Utility System

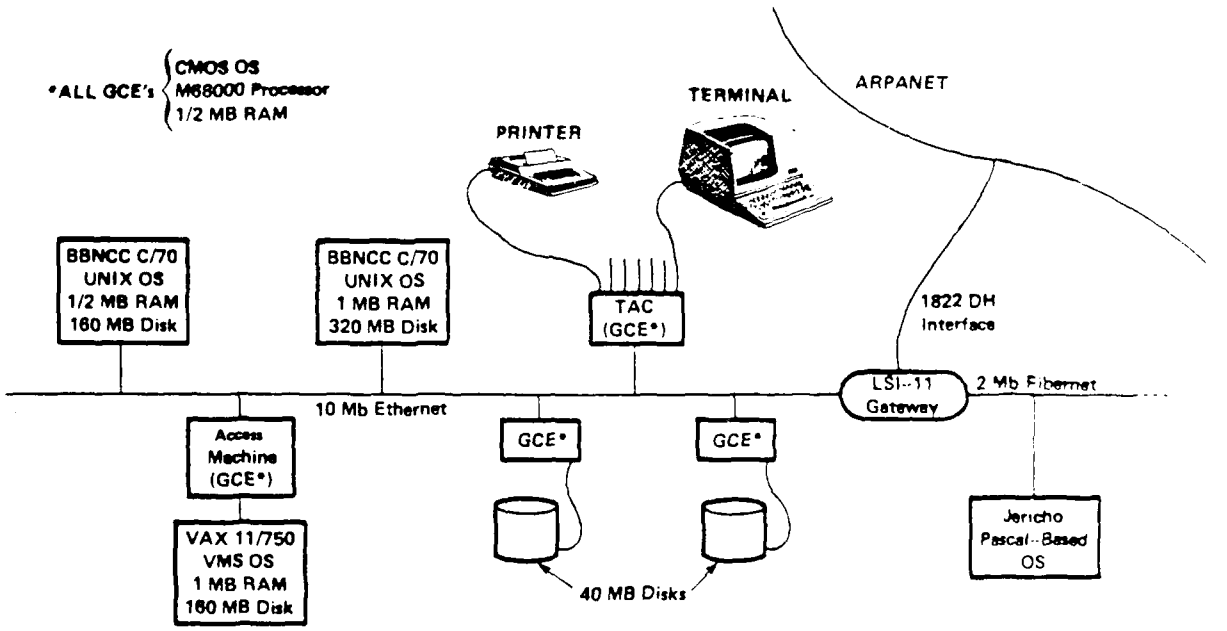
The utility system consists of the minimal system plus one or more fully-integrated, general-purpose timesharing hosts called utility hosts (the C/70 computers will play this role in the Advanced Development Model) The utility system will be suitable for developing new applications in the framework of the DOS, and will support the utilities typical of a modern timesharing system The utility system will also support the

maintenance of its own software and the software of the minimal system

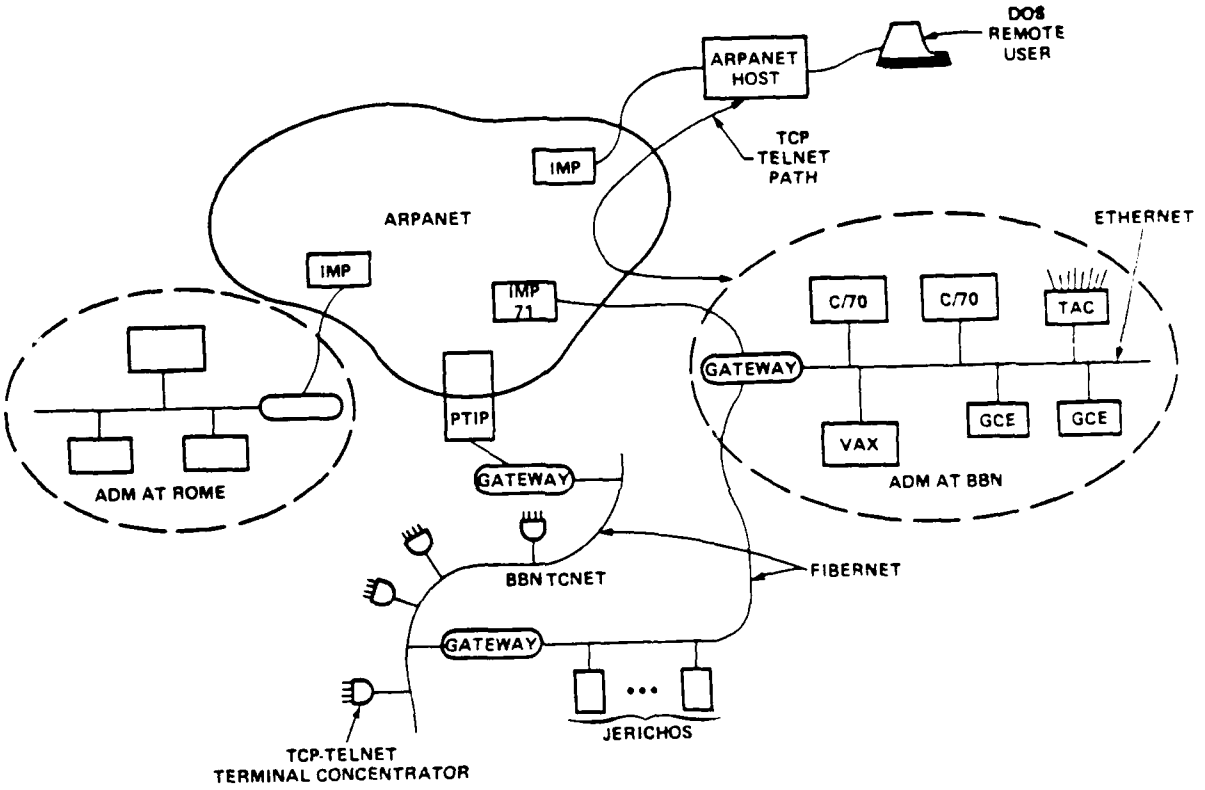
2.5.3 Level 3: The Application System

The application system consists of the minimal system and some number of application hosts, workstations, and special-purpose devices. An application system may simultaneously be a utility system, if utility hosts are present in the cluster. Applications are generally developed in a utility system and operate in an application system. Application systems, therefore, need not be capable of supporting their own software development. Application systems are sometimes configured with redundant components and operated in a high reliability mode. Note that GCE's can be used for application programming, thus a particularly simple application system consists of just the network, the GCE's required to provide essential services, and some number of application GCE's.

Figures two and three illustrate the components and the context of the initial system configuration for the Advanced Development Model being assembled at BBN.



The Local Cluster Configuration
Figure 2



The InterCluster Environment
Figure 3

3 The DOS Functions and Underlying Concepts

3 1 Introduction

Expected usage of the DOS can be divided into five categories

1 Applications.

2 Application development and maintenance.

3 System administration.

4 System operation.

5 System development and maintenance.

The system is intended primarily to support end application usage (1). However, to adequately support end applications it must also support the other categories of use. Therefore it should be possible for users working in each of these cases to perform their responsibilites by means of the DOS. The goal of supporting these usage categories places requirements on the functions the DOS must implement, and on the tools it must be able to accommodate. This section discusses the DOS functions.

The DOS system will provide functions in the following areas.

- System access. The objective is to support flexible convenient access to the system from a variety of user access points.
- Object management. The notion of a "DOS object" is central

PREVIOUS PAGE IS BLANK

to the user model for the DOS The DOS treats resources, such as files, programs and devices, as "objects" which it manages, and which users and application programs may access The objective of the object management mechanisms is to provide users and application programs uniform means for accessing DOS objects

- Process management Like the object abstraction, the "process" abstraction is central to the user model of the DOS In addition, it is useful as an organizing paradigm for the internal structure of the DOS The objective of the DOS process management mechanisms is to implement the "process" notion in a way that enables processes to be used both to support the execution of application programs for users and internally to implement DOS functions

- Authentication, access control, protection, and security The objective is to provide controlled access to DOS objects

- Symbolic naming DOS users will generally reference objects and services symbolically Symbolic access to DOS objects will be supported by means of a global symbolic name space for objects

- Interprocess communication The objective of the interprocess communication (IPC) facility is to support communication among processes internal to the DOS, and among user and application level processes

- User interface The user interface functions provide human users with uniform, convenient access to the features and services supported by the DOS resources

- Input and Output The objective here is to provide flexible and convenient means for users and programs that act on the behalf of users to make use of devices such as printers, tape drives, etc

- System monitoring and control The purpose of the system monitoring and control functions is to provide a uniform basis for operating and manually controlling the system

The principal goal for the DOS in each of these functional areas is to support features that are comparable to those found in modern, conventional, centralized operating systems such as

Unix, Multics, VMS, and TOPS-20

The development of radically new types of operating system functions and concepts, except for those required to deal with the distributed nature of the system, is not a major goal of the DOS effort. This position is motivated by two considerations

1. It is important to avoid innovation in too many areas when building a system. The important innovations embodied by the DOS will result from addressing problems posed by distribution. These problems span the functional areas identified above. Therefore, most of the effort must be directed toward making the system operate in a coherent, survivable and efficient manner in a distributed environment rather than toward developing new operating system concepts

2. However, unless the functions provided are comparable in power and convenience to those found in centralized systems, users will not choose to use the DOS. Thus it is important for the success of the DOS as a system that it provide state-of-the-art capabilities

The rest of this section discusses the functional areas identified above in terms of our objectives in each area, and sketches some of the concepts and principles that underlie our approaches for achieving the objectives

Each functional area is discussed in a separate section. However, it will become clear from the discussion that these functions are not independent of one another. These interrelationships occur across functional areas as well as within them. For example, objects and processes are intimately

interrelated A process is a type of DOS object and access to DOS objects is supported by interactions among processes Furthermore, internally the system is structured to combine lower level functions and capabilities in one or more areas into higher level functions and capabilities. For example, the relatively higher level notion of reliable (multiple copy) file objects is implemented by more basic (single copy) file objects

This internal "involuted" structure of the system is important. If the structure and interrelationships are designed well, implementation can proceed in orderly and efficient stages from the lower levels to the higher ones. Furthermore, the resulting system implementation will exhibit internal order, making it easier to maintain and to modify for adapting to new requirements

3.2 System Access

The objective in this area is to provide users with flexible, convenient access paths to the system

The system will support a number of different types of access points including

1 Terminal access computers (TACs) A TAC is a terminal multiplexer connected directly to the DOS local area network It acts to interface a number of user terminals

to the DOS. The software that runs on a TAC is entirely under the control of the DOS. User programs are not permitted to run on a TAC computer.

2. Dedicated workstation computers. A workstation is a computer that is, at any given time, dedicated to a single user. Workstations will be connected to the DOS local network. Workstation hosts have sufficient processing and storage resources to support non-trivial application programs, such as editors and compilers, and to operate autonomously for long periods of time. A workstation may serve as its user's access point to the DOS. User programs may run on a workstation.

3. The internetwork. The DOS local network is connected to the internetwork by means of a gateway computer which is a host on the DOS local area network. Users remote from the DOS cluster may access the DOS through the internetwork. Remote terminal access is accomplished by means of a standard terminal handling protocol (TELNET) which operates upon a lower level, reliable transport protocol (TCP).

Because of the distributed nature of the system, user interaction with the DOS is supported by software that runs on one or more computers. This software includes two principal modules. One module is responsible for handling the user's terminal. Since this module will often run at or very 'near' the user's access point, we shall call it as the "access point agent". The other principal user interface module interacts with the user at a higher level to provide access to DOS resources in response to various user commands. We shall call this module the "user agent". It is useful to think of the access point agent and the user agent as processes. These agent processes interact with other components of the DOS and with each other by means of well defined interfaces and protocols. In addition, they play an

important role in insuring the reliability of user sessions

The access point for a user session, in part, determines where the access point agent and user agent processes run For a user whose access point is a TAC the access point agent runs on the TAC, and the user agent runs on a shared host. The access point agent for a user with a dedicated workstation runs on the user's workstation computer, and the user agent may run on the workstation or it may run on a shared host. Users who access the DOS through the internetwork are allocated user agents that run on shared hosts, and their access point agents may run either on the (non-DOS) host used to access the DOS or on a host within the DOS cluster.

Some DOS hosts may provide support for terminals directly connected to them. It will be possible for users to access the DOS through such directly connected terminals These users will be treated much like users who access the DOS through the internetwork in the sense that the DOS will allocate user agents for them that run on shared hosts

The standard user interface software (for users accessing the DOS through TACs and the internetwork) will be written to operate with CRT terminals that have cursor positioning capabilities. In particular, this includes terminals that meet a subset of ANSI standards X3.41-1974 and X3.64-1977, providing

cursor positioning and various other functions such as clear to end of line, delete line, insert line, etc. More capable terminal devices (e.g., workstations with graphics displays) can emulate the standard terminal device to obtain a compatible user interface. In addition, a means will exist for users with other less capable terminal devices (e.g., printing terminals) to access the system (e.g., by using the TELNET Network Vitual Terminal or NVT as a lowest- common denominator terminal device). In the latter case, some sacrifice in the quality, uniformity, and power of the user interface is unavoidable. The user interface is discussed further in Section 3.8.

3.3 Object Managememt

The DOS will support a wide variety of objects. The objective of the DOS object management mechanisms is to provide access to DOS objects.

DOS object management will be based on the following principles.

- Every DOS object has a unique identifier. At the lowest level within the system, access to a DOS object can be accomplished by specifying its unique identifier and the desired access to an "object manager" process for the object.
- The DOS will support a collection of transaction-based

object access protocols. These protocols will be type dependent in the sense that there will be different access protocols for different object types.

- Access to objects will be accomplished by engaging in the appropriate access protocol with an object manager process for the object. The interactions between the accessing agent and the object manager will be accomplished by means of interprocess communication (See Section 3.7).

- Input/output devices will be treated as DOS objects. Consequently, input/output devices will have object managers, and access to the devices will be accomplished by means of interprocess communication.

- The DOS catalog (See Section 3.6) provides a means of binding symbolic names to DOS objects. The catalog supports a lookup function (a symbolic name-to-unique id mapping) which enables objects to be accessed symbolically.

- The DOS will support a fixed set of basic object types (such as "primitive" file, "primitive" process, etc.). In addition, it will support more complex object types (such as "multiple copy" file, "migratable" file, etc.) which will be built upon the properties of the basic object types. Our design objective at this time is to develop the framework for supporting more complex object types, rather than to try to specify the semantics of those object types.

Files are a particularly important type of DOS object. The storage resources of dedicated DOS hosts as well as certain constituent hosts will be used to store DOS files. Symbolic naming for DOS files will be implemented by the DOS catalog.

Each host that provides storage for DOS primitive files will also support the object manager which implements the DOS access protocol for primitive files.

3.4 Process Management

As suggested above, the DOS will support the notion of a process. Processes will be used both by the implementation of the DOS and to directly support user applications. For example, there will be processes responsible for implementing the DOS object catalog and for implementing the DOS file system. In order to support user processing activity, there will be processes that execute standard tools, such as text editors and language processors, as well as specific command and control applications.

The objective of the DOS process structure mechanisms is twofold:

1. To support the process concepts required to implement DOS functions, for example, object management.
2. To provide a basis upon which to develop means for users to initiate and control processing activity within the DOS.

DOS process management will be based on the following principles:

- A basic type of process ("primitive" processes) will be implemented at a fairly low level. It will be bound to a particular host, and it will bear no special relationships or capabilities with respect to other primitive processes.
- Primitive processes are DOS objects. As such, they have unique identifiers, and may be cataloged in the DOS catalog. (See Section 3.6). So called "server" processes that provide services useful to a wide range of clients are

examples of processes which are useful to catalog Cataloging such a process enables it to be referenced symbolically by the general population of client processes

- More sophisticated process notions will be built upon the primitive process notion For example, the notion of hierarchical process structures, where processes are related to one another according to the manner in which they were created, and where the relationship between processes determines the types of operations a process can perform on other processes, will be built upon the primitive process notion Similarly, "migratable" processes (processes that can move from one machine to another) will be built upon primitive processes

- The system will support the notion of "long lived" processes. A long lived process is one which the system will take steps to ensure exists over shut downs and restarts of the system and of individual hosts. Server processes will frequently be long lived

- Process i/o and interprocess communication will be handled in an integrated fashion. The notion of "primary" input and output streams for a process will be supported, and it will be possible to "link" processes together by connecting the input stream of one process to the output stream of another. Among other things, this will make it possible for one process to act as a filter or translator for the stream of data passing between two other processes

3.5 Authentication, Access Control, and Security

The objective of the DOS in this area is to provide for controlled access to DOS objects The purpose of the DOS access control mechanisms is

1 To prevent the unauthorized use of DOS objects For example, it is important to ensure the privacy of sensitive data by preventing unauthorized users from accessing it

2 To ensure the integrity of DOS objects The objective here is to control the ways in which various objects may be used

Convenient and flexible means should be available to users for specifying the types of access other users may have to their objects

The access control mechanisms will be designed to be strong enough to protect the privacy and integrity of DOS objects against accidental disclosure or misuse, and against attacks by malicious, but inexpert users. It is extremely difficult to protect against attacks by dedicated expert users, and it is not a primary goal for the DOS to be invulnerable to such attacks

There are two capabilities related to protection and security that are not goals for the DOS

- Prevention of denial of service Denial of service occurs when a user prevents or interferes with someone else's use of the system or parts of it A simple example would be a user who seizes all the "job slots" on a timesharing system by logging in many times, thereby preventing others from accessing the system Another example would be the situation that might occur if a user could run a program that floods the local area network with packets This would prevent other users from using the network Although the DOS will be able to prevent certain types of denial of service, including those just described, it is very difficult, in general, to comprehensively prevent denial of service
- Implementation of the military security model The DOS will not implement multi-level security The DOS would run in a "system high" mode if it were used to process classified data The DOS access control mechanisms could be used, however, as a support for the Need-To-Know

security model, just as access control in commercial single-host operating systems is used for this purpose

Internally, the DOS will be organized so that much of its operation is accomplished by means of processes Many of these internal DOS processes may be thought of as agents which act to carry out user requests The principal DOS access control mechanism will be based on the identity of the agent attempting to access an object An important part of access control procedures within the DOS will be to determine the identity of the accessing agent and the identity of the user on whose authority the agent is acting Consequently, reliable authentication of users and processes will be an important element of the DOS access control mechanisms

The DOS protection and security mechanisms will be based on the following principles

- Each DOS user will have his own unique identity which is understood across the entire DOS system
- Users of the DOS will be required to login once per user session In most cases access to DOS resources during a session will not require additional logins that involve explicit user participation
- User login will be accomplished in the conventional manner by supplying a valid user login name and password
- User passwords that are stored within the system will be protected by means of a one-way (i e , non-invertible) transformation A password check will be performed by first applying the transformation to the password supplied by a user and then comparing the result with the transformed password for the user that is stored by the

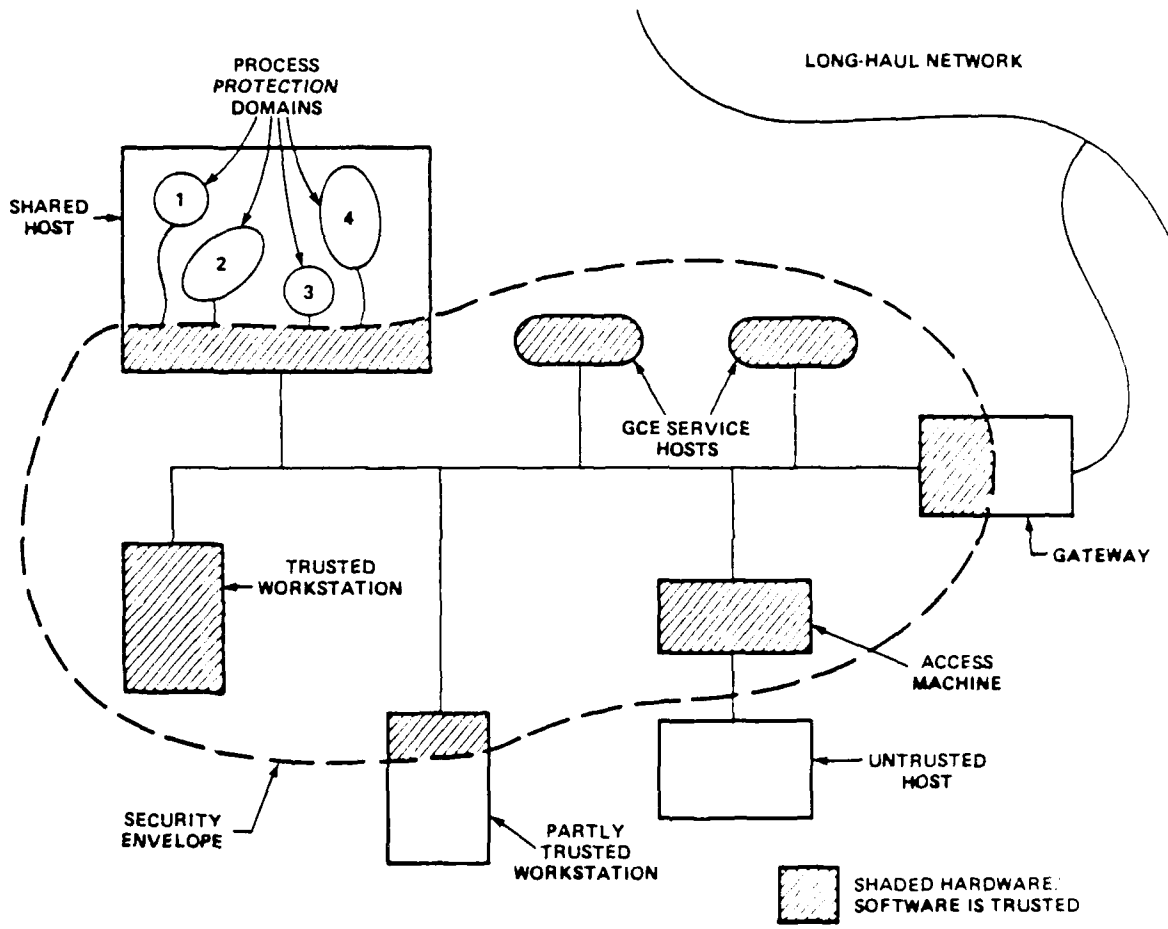
system

- All attempts to access DOS resources will be subject to access control checks prior to access

- All attempts to access DOS objects, including those initiated from access points which are external to the DOS will be treated by the system as being made on behalf of some registered system user. In order to enforce the appropriate access controls the object managers for DOS resources must be able to obtain the identity of the registered user from the accessing agent or to determine it from information supplied by the accessing agent

- We assume the existence of a "security envelope" which surrounds the DOS local area network and some of the key DOS components (see Figure 4). The security envelope protects the network in the sense that access to the network is controlled. This control is accomplished by means of physical security, system hardware, and DOS software. Unauthorized users (or hosts) are not free to listen to communication on the network, and are not free to send arbitrary packets. DOS components which are within the security envelope may trust each other, and processes outside of the security envelope are not able to masquerade as trusted processes.

Figure 4 shows the possible relationships between hosts and the security envelope. A shared host (typically a timesharing application host) will participate in the DOS access control mechanisms by means of augmentation to its trusted "monitor" or "supervisor" processes. Generic Computing Elements which supply DOS essential services will be wholly contained within the security envelope (i.e., untrusted applications are not permitted to directly alter the programs resident in system GCE). Gateways attached to the cluster must penetrate through the security envelope because they connect the trusted network to the untrusted internet. At a minimum, gateways must apply [illegible]



The DOS Security Envelope
Figure 4

mark all traffic entering the cluster as "foreign" in a trustworthy manner Access machines may be used to connect completely untrusted hosts to the cluster In this case the access machine would validate all interactions between the untrusted host and the DOS components inside the security envelope Workstations attached to the DOS may either be fully trusted, and hence inside the boundary of the security envelope or partially trusted A partially trusted workstation is presumed to contain some tamper-proof hardware and software components that protect the DOS from anti-social behavior on the part of the workstation.

It is desirable to provide means for a user of the DOS who has the ability to access a particular object to pass (perhaps limited) rights to access that object outside of the DOS cluster This would enable a user of the DOS to permit others who are not registered DOS users to access specific DOS objects in a controlled fashion. The absence of this feature on ARPANET hosts is a considerable impediment to sharing across host boundaries

This will be accomplished by a mechanism which will enable a DOS user to create a "capability" for a particular object (e g the ability to read a certain file) and then pass the capability on to someone else When a request to access an object is accompanied with the capability for the object, the DOS may grant

access to the object after checking the validity of the capability. To ensure that this feature does not compromise the privacy and integrity of DOS object capabilities must be such that they cannot be forged. To help ensure that registered DOS users can be held accountable for their actions it is desirable that the DOS be able to deduce the identity of the user who created a given capability.

3 6 Symbolic Naming

Naming is an important unifying concept for the DOS. The means provided for naming objects is one of the most important factors determining how easy and convenient a system is to use.

The DOS will implement a global symbolic name space for DOS objects. This name space will have the following properties:

- The symbolic name for an object will be independent of the object's location within the DOS.
- The symbolic name used to refer to an object will be the same regardless of the location within the DOS that the name is used.
- Common syntactic conventions will apply to symbolic names for different types of objects (including files, devices, server processes, etc.).

The symbolic name space will be implemented by means of a DOS catalog data base (or simply "catalog"). The catalog will

implement a symbolic name-to-object mapping for the DOS objects it catalogs. The catalog will not usually store the objects themselves, but rather will store information about the objects. Information about an object will be stored in a catalog entry for the object. This information will be sufficient to allow access to the object. In particular, the catalog will store the global unique identifier for each object it catalogs along with any additional information required to locate the object within the DOS. In addition, it will also maintain certain attributes of objects it catalogs.

While in some sense the catalog can be thought of as a logically centralized data base, it will be implemented in a distributed fashion. In particular, the catalog will be dispersed among a number of DOS hosts and some parts of it may be replicated. It will be dispersed to ensure that the system is scalable and that the catalog is reliable. While all of the information in the catalog, even for very large configurations might fit on a single DOS host, it seems unwise to store it on a single host. In large configurations the load placed on that host would likely represent a performance bottleneck. Furthermore, the cataloging functions would be vulnerable to a failure of that single host. Parts of the catalog will be replicated to ensure high availability of critical catalog data.

The symbolic name space and its supporting catalog will be based on the following principles

- The name space will be hierarchical The name space hierarchy can be thought of as a tree with labeled branches.
  - o The leaves (terminal nodes) of the tree represent cataloged objects
  - o The symbolic name for an object is the name of the path from the root node of the tree to the node that represents the object.
  - o Non-terminal nodes of the tree represent collections of catalog entries and are called "directories"
  - o Directories are DOS objects, and they have names The name of a directory is the name of the path from the root to the node that represents the directory Thus, the non-terminal nodes of the tree also represent cataloged (directory) objects
- A set of general operations for manipulating the catalog, directories and catalog entries, independent of the types of objects, will be provided.
- The catalog can be used to obtain information about an object However, issues associated with accessing the object, such as access protocols and object representation are separate from the naming issues that are addressed by the catalog
- The catalog data base will be organized to efficiently implement two types of lookup operations symbolic name-to-catalog entry, and unique id-to catalog entry The symbolic name lookup operation is supported for human users "Wildcard" designators will be supported The unique id lookup operation is supported for programs
- Operations which modify the catalog will be implemented as atomic transactions in order to maintain the integrity of the catalog in the presence of concurrent activity and possible failures of system components
- The catalog will have the ability to maintain linkages to

other name spaces. This is supported to permit name spaces of constituent hosts to be (weakly) integrated into the DOS symbolic name space. This will be accomplished by an "external name space" object which can be cataloged like any other object. For example, it will be possible to catalog the directory /usr/rjones/memos on some Unix DOS host as a DOS external name space object. Coupled with appropriate file access software on the Unix system, this would permit a user to refer directly to files in the cataloged directory from the DOS name space.

- The catalog can be thought of as a (complex) DOS object. As mentioned above, directories within the catalog are DOS objects. Therefore, access to the catalog can be controlled by the same mechanisms that control access to other DOS objects. This access control will help ensure the privacy and integrity of information in the catalog. Access to the objects themselves are, of course, also subject to access controls.

- Components of the DOS may choose to cache catalog entries or the contents of entire directories, in order to support lookup operations locally. This would be done to avoid the potential overhead associated with interacting with remote catalog data bases.

The catalog is an important component of the DOS. It will be used not only to support the cataloging requirements of DOS users, but also to support the implementation of parts of the DOS. For example, as noted above in Section 3.3 the symbolic naming requirements of the DOS file system will be supported by the DOS catalog.

Not all DOS objects will be cataloged in the catalog. It will be possible to access uncataloged objects directly by means of their unique ids.

3.7 Interprocess Communication

The objective of the DOS interprocess communication (IPC) facility is to support the communication requirements of the DOS. Requirements can be identified at two levels:

1. The system implementation level. The collection of software modules that implement the DOS execute as processes on various DOS hosts. These processes must interact to implement the DOS. These interactions are supported by the interprocess communication facility.

2. The user application level. Some of the application programs that execute in the DOS environment may be structured as distributed programs. A distributed program is one whose components may run as cooperating processes on different hosts. The components of such a distributed application program will need to communicate.

The IPC facilities that are available at the application level will be built upon the system level IPC facility.

The DOS interprocess communication facility will be based on the following principles:

- The IPC mechanism will support a variety of communication modes including datagrams and connections (i.e., reliable sequenced, flow controlled data streams).

- It will be built upon the standard DoD IP (internet) and TCP (transmission control) protocols. This assumes that the implementations of the DoD protocols that are used will provide adequate performance (low delay, high throughput). If they do not, it may be necessary to build the IPC directly on the local network (Ethernet) protocol.

- Interhost and intrahost communication will be treated in a uniform fashion at the interface to the IPC facility. That is, the same IPC operations used for communicating with processes on different hosts will be used for communicating

with ones on the same host. Of course, to achieve the efficiencies that are possible for local communication the IPC implementation will treat interhost communication differently from local communication.

- Several levels of addressing will be supported by the IPC facility. The details of IPC addressing within the DOS have not yet been finalized. The fundamental issue which is unresolved is what the addressable entity for the IPC facility shall be, that is, to what will datagrams be addressed and what will connections connect? One reasonable choice would be for the process itself to be the addressable entity. Alternatively, another abstraction the "port", could be introduced for this purpose. Ports would be objects, and like other objects such as processes they would have unique ids and, if cataloged, could be referenced symbolically by name. Regardless of the choice for addressable entity, the IPC facility will permit addressing by means of unique id and by means of symbolic name. Other levels of addressing may also be supported. At the interface to the IPC facility wherever an IPC address is expected, any of the supported levels of addressing (unique id, symbolic name) may be used.

- The ability of the IPC facility to deal with symbolic addresses will permit it to support "generic" addressing. This will permit processes to specify interactions with other processes in functional terms.

- The IPC mechanism will provide means to directly utilize some of the capabilities of the local network. For example, the Ethernet supports efficient broadcast and multicast. The IPC will provide relatively direct access to these capabilities by supporting broadcast and multicast addressing. To achieve the design goal of component substitution it is important for the DOS system to be as independent as possible of the specific characteristics of the particular local network chosen for the ADM. Therefore, care must be taken to avoid building dependencies on the particular ADM network technology into lower level DOS mechanisms, such as the IPC. If such dependencies cannot be avoided, care must be taken to minimize their impact on the DOS. In our opinion, this is not an issue in the case of the broadcast and multicast facilities, since many state-of-the-art local network technologies support similar capabilities.

3.8 User Interface

The purpose of a user interface to the DOS is to provide human users with uniform, convenient access to the functions and services performed by the DOS resources.

The user interface is software that acts to accept input from a human user which it interprets as commands to perform various tasks and to direct output to the user which the user interprets as the results of commands previously requested or as unsolicited information from the system (or possibly other users). As discussed in Section 3.2, it is sometimes useful to think of the user interface functions as being provided by access point agent and user agent processes.

"Uniform" and "convenient" are subjective characteristics which are hard to quantify. However, we can say in general terms what we mean by these characteristics in the context of a DOS user interface. By uniform, we mean that the manner in which a user requests access to various functions and resources should be similar regardless of the particular DOS components that implement them. For example, the way a user instructs the DOS to run a program should be the same (except for the name of the program) regardless of where within the DOS the program will execute. By convenient, we mean that a user should not have to pay undue attention to the details of the mechanisms of

establishing access to DOS functions and resources For example in order to run an interactive program, a user should not have to explicitly establish a communication path with the host that will run the program Similarly, to delete a file a user should not have to explicitly establish communication with a file manager on the host that stores the file and instruct it to delete the file

To be uniform and convenient does not mean that a user interface must make the network or the distribution of the system invisible to users In many situations users may want the distribution to be transparent, and the user interface should operate in a way that provides transparency However, there will be situations where it will be important for the distribution to be visible to users, and for users to be able to exert control over how the system deals with aspects of the distribution For example, to use the system to do their jobs system operators and maintainers will need to deal relatively directly with the system's distributed nature Furthermore, "normal" users from time to time, may want to control where programs run or files are stored

One of the ways the DOS will differ from most conventional single host operating systems is that truly parallel execution of user tasks will be possible It is important that a user interface for the DOS provide means for to initiate, monitor and

control multiple concurrent tasks

The development of DOS user interface functions will be based on the following principles, many of which are particularly well suited to interactive command and control environments

- Since many user requests cannot be performed directly by the user interface, the user interface acts on the user's behalf to initiate activity by other DOS modules The nature of the interactions with other DOS modules is governed by internal DOS "protocols" and interface conventions, and is accomplished by means of interprocess communication

- An important type of activity a user can initiate is the execution of a program In this case, the user interface acts to initiate execution of the program and to establish a communication path between the user and the program In addition, means are provided to permit a user to switch his attention back and forth between the executing program and the user interface

- The user interface will enable a user to initiate and control multiple simultaneous tasks In particular, a user may have several application programs executing concurrently.

- Although the user interface bears a unique relationship to the rest of the DOS system, the underlying DOS system will be organized so that much, if not all, of the user interface functions can be written as application level software

- The part of the user interface that interacts directly with the user to accept commands will be modularized in a way that allows it to be replaced on a per user basis At login time, after the user is identified, the particular user interaction module appropriate for the user will be used This will make it possible to accommodate users with strong preferences for radically different styles of interaction, simply by running different user interaction modules

- The user interface functions developed for the ARM [ref] will

be designed to operate best with a high speed CRT display terminal, with cursor positioning capability (See Section 3.2). It will make use of multiple "windows" on the display surface. Separate windows will be used to display user interactions with the separate activities being controlled by the user. In addition, windows will be used as necessary to display system status and user help information. The ADM user interface will be tailorable to accommodate a relatively broad range of individual user preferences. This will be accomplished by means of a number of internal "style" and "mode" parameters whose settings control the way the user interface performs. The settings for these parameters will be initialized from values stored in individual user profiles and will be able to be modified at the user's request during a user session.

3.9 Input / Output

The term "input/output" is used here in a rather limited sense to mean the process of getting data into and out of the DOS cluster. The objective of the DOS in this area is to provide flexible and convenient means for users and application programs to make use of devices such as printers, tape drives, etc.

To support I/O adequately in its distributed environment, the DOS should provide:

1. The ability to refer to devices symbolically. For example, users should be able to obtain listings of files by means of "print" or "list" commands which explicitly or implicitly refer to a printer symbolically. Similarly, programs should be able to direct output to a printer by referring to it symbolically.

2. The ability to distinguish among and to refer to physical devices. In moderate and large configurations, there will be more than one printer (or tape drive, etc.). The

devices are likely to be located in different areas It is critically important that the tape drive from which a program reads is the one that holds the right tape Similarly, when a user requests a listing it is important for him to be able to control which printer will print it so that the output is near his office rather than 1 ½ mile away Thus, one user s "printer" will not necessarily be the same as another's Furthermore when a user accesses the DOS from a different location then normal, he should be able to rebind his "printer" to one of the printers that are near him

The object paradigm developed above, which involves objects, object managers, and object access protocols, is almost sufficient to support DOS device i o In addition, the system will provide means for a user to "bind" a particular symbolic device name to a particular physical device

In summary, DOS support for i o will be built upon the following principles

- Input output devices will be treated as DOS objects As such, they will have unique ids and may have symbolic names
- Access to devices will be supported in the same way access to other DOS objects is supported Access will be accomplished by interacting with an object (device) manager in accordance with an appropriate object (device) access protocols The interactions will be supported by means of interprocess communication
- The notion of device binding will be supported by means of the DOS catalog This will permit users to bind symbolic names to particular physical devices
- Some types of i o operations when suitably abstracted are meaningful for files and for devices Sequential i o is a good example File-like interfaces for devices have been shown to be useful in a number of systems The DOS will support file-like interfaces for certain devices

3 10 System Monitoring and Control

The purpose of the DOS system monitoring and control functions is to provide a basis for system operations personnel to operate and control the system

The system monitoring and control functions will be built upon the following notions

- Two types of information will be gathered system status information, and information about the occurrence of exceptional events Status information will be collected on a periodic basis as a normal part of system operation Information about exceptional events will be collected as the events are detected
- Status information and information about exceptional events will be routed to an on-line display which system operations personnel can monitor
- The detection of certain exceptional events will trigger an "alerting" mechanism to call the events to the attention of operations personnel
- It will be possible to (selectively) log the occurrence of exceptional events in a event log data base
- The DOS will support a system control protocol which will make it possible for operations personnel to control the system operation from a single point (e g operator's console) as a DOS user This protocol will provide means to reinitialize the system ("warm" restart) to halt the system, and to set parameters within various DOS modules which control aspects of the DOS operation
- The status gathering facilities will be flexible and comprehensive enough to support performance measurement experiments

3.10 System Monitoring and Control

The purpose of the DOS system monitoring and control functions is to provide a basis for system operations personnel to operate and control the system.

The system monitoring and control functions will be built upon the following notions:

- Two types of information will be gathered: system status information, and information about the occurrence of exceptional events. Status information will be collected on a periodic basis as a normal part of system operation. Information about exceptional events will be collected as the events are detected.
- Status information and information about exceptional events will be routed to an on-line display which system operations personnel can monitor.
- The detection of certain exceptional events will trigger an "alerting" mechanism to call the events to the attention of operations personnel.
- It will be possible to (selectively) log the occurrence of exceptional events in a event log data base.
- The DOS will support a system control protocol which will make it possible for operations personnel to control the system operation from a single point (e.g., operator's console) as a DOS user. This protocol will provide means to reinitialize the system ("warm" restart), to halt the system, and to set parameters within various DOS components which control aspects of the DOS operation.
- The status gathering facilities will be flexible and comprehensive enough to support performance monitoring experiments.

4 System Integrity and Survivability

Users of modern day computing facilities have come to expect the integrity of their computing system and the data it stores and manipulates for them, despite occasional system component failures. The command and control environment in particular requires the continuous availability of key applications despite these failures. To the extent that applications and access to applications come to depend on DOS system functions to achieve goals of system uniformity, those functions must be reliable and continuously available. Further, the role of the DOS as the common software base extending throughout the cluster, makes it a convenient and cost-effective place (from a programming standpoint) to support generalized system wide mechanisms for building survivable applications.

By availability we mean the fraction of scheduled up-time during which a system is, in fact, able to deliver normal services to its users. Continuous availability then refers to the ability of the system to supply services without pause over some relatively long period of time. The period is sufficiently long to present a significant chance of component failure. Thus a system design which achieves continuous availability must employ some elements of fault-tolerant system design. By integrity we mean the operation of the system in accordance with

PREVIOUS PAGE IS BLANK

its specifications while it is available, despite failures from time to time which may render the system temporarily unavailable Maintaining system integrity is basically a matter of maintaining the consistency of system and user state information ("stored data") The term survivability is virtually synonymous with "continuous availability", although the emphasis is perhaps different, "survivability" suggesting the possibility of violent failure modes.

A goal of high (but not continuous) availability implies attention to mechanisms for orderly system restarts, that will preserve system integrity across system outages The restart process may be partially manual, and may involve relatively lengthy integrity checks and system reconfiguration procedures (e g , replacing a disk pack, restoring files from backup tapes) Continuous availability, in our terminology, refers to the ability of the system to automatically reconfigure itself or to retry failed operations, in order to maintain the normal semantics of a given function in spite of failures In a continuously available (i e , survivable) system, a failure manifests itself only as a tolerable performance degradation and or insignificant loss of data or function

Our distinction between high and continuous availability can be illustrated by the following examples Operator invoked

reversal to a backup copy of a damaged file would constitute a recovery measure suitable for a goal of high availability. In contrast, designing a function (e.g., authentication service) so that the system can automatically detect a host failure and subsquently route requests to an alternate source of the function, would be a mechanism for continuous availability. In either case, the integrity of the system must be maintained whenever system services are available.

At a minimum, key system functions and applications must be highly available, and in many cases also continuously available. Ideally, all system services would be continuously available in the command and control environment. However, cost and performance criteria may dictate that high availability is acceptable for some functions, especially if the expected failure rate is low. Functions such as authentication, initiation of user sessions, and access control must be continuously available for the system to operate at all. Other functions (e.g., access to selected application data) may satisfactorily be provided on a highly available basis, whereas still other functions (e.g., data collection for experimentation) need not be provided at all, unless all system resources are operating normally.

All three aspects, integrity, high availability, and continuous availability, play important roles in the overall

effectiveness of the system for command and control environments and will be collectively referred to as system reliability.

4.1 Reliability Objectives

The reliability objective of an automated command and control cluster is to provide reliable command and control applications. The role of the "system" with respect to the reliability of these applications is threefold.

- Ensure the "correct" operation of the system in the presence of expected patterns of component failure and subsequent restorations of service. Included in this is that the system does not, under a broad range of failures lose or corrupt data that is essential to either its own "correct" behavior or to the "correct" behavior of its supported applications.
- Provide key DOS system functions and access to those functions in a manner which can survive a limited set of system failures, and which is designed to support high availability.
- Provide DOS based mechanisms accessible at the user programming interface which are useful for constructing reliable applications.

4.2 General Approach

Our approach to failure handling in the DOS is based on first identifying the set of failure modes over which the system is expected to maintain integrity and be continuously available.

The definition of each major DOS system function includes the integrity and survivability characteristics to be supported should the expected failures occur. Based on the reliability properties of the specific system functions, other functions using them can then be built which are immune to the outages handled by the abstract function.

The integrity and consistency of system functions are derived from the careful ordering and synchronization of the parts of the individual and parallel operations, and the grouping of related parts into atomic operations that have coordinated outcomes. DOS functional survivability always derives from redundancy of one form or another, either in processing elements and executable programs, or in data, or in time (operation retries). Making the data accepted for storage by the system resilient to component and storage media failures, in the sense that data is not lost despite these failures, is one special case of the general redundancy concept.

The DOS architecture calls for hardware redundancy to support all survivable functions. The approach is to provide a homogeneous processing base for each particular survivable function, as a means of simplifying the issues of fidelity and coordination between the redundant elements. The role of the DOS software is to support the replication of critical code and data

to control the detection of failures, and to induce recovery procedures In some cases, such as transaction processing multiple redundant servers will be supported to share the processing load in the absence of failures, as well as to provide continued service during failures In other cases such as data processing application survivability, restart from a prior consistent checkpointed state represents a powerful base on which to build In all of these cases, the presence of a homogeneous processing base is essential in limiting the complexity of implementation.

4 3 Specific Approach

We expect the key functions of the the DOS to be able to recover from the following types of failures

- Single host outage at arbitrary time without loss of non-volatile memory This comes in two forms, transient in which the host is restarted within minutes, and long term (hours at minimum) during which the host is effectively no longer available Transient failures of this sort are expected frequently (a few times per day for large configurations) while long term failure is relatively infrequent (a few times per month)

- Single host outage at arbitrary time with additional loss of long term non-volatile memory (e g disk crash) These failures are always long term, and occur infrequently (a few times per year)

- Operator controlled forced host shutdown with ample warning for proper shutdown preparation (e g down for emergency or preventive maintenance) This occurs

relatively frequently (a few times per week)

- Transient pair wise communication failures This is predominantly a temporary failure, with the expectation that subsequent retries over a sufficiently long interval will succeed This condition frequently occurs due to temporary congestion, random noise, hardware and software interfaces not designed for worst case timing conditions, etc

- Single host temporarily loses communication with the rest of the system but continues to operate This is the long term version of the pair wise transient communication failure pattern, across all pairs for this host It occurs relatively infrequently and can be the result of a malfunctioning network interface This single host isolation represents the most likely pattern of network partitioning which can be anticipated using current local communication bus architectures.

- Any failures that can be made to look like one of the above.

In general, handling failures involves techniques for failure detection, reconstitution of remaining components into a working system, and subsequent reintegration of temporarily failed components back into the operational system after they are repaired The techniques selected to detect and recover from these failures will vary depending on the expected duration and relative frequency of the failure Mechanisms selected to handle infrequent events can usually be of limited performance and include manual procedures Mechanisms for frequently occurring events must also take into account the performance characteristics the solutions adopted

The following techniques have been well studied and are

suitable for supporting various aspects of system reliability in the DOS [1]

- Redundancy of program, file, and processing elements as sources of alternate site service.
- Atomic operations and isolation of partial results to ensure the consistency of function and data.
- Stable storage and guaranteed permanence of effect to ensure that data and decisions once accepted by the system, will not be lost.
- Checkpoint and restart to support backward error recovery
- Timeouts to recognize failure conditions and initiate recovery activities.
- Status probes and status reporting to ensure current operability

In addition, the GCE concept of interchangeable parts is viewed as a manual approach toward easily reconfiguring components for continued support of important system functions by using parts from less important functions utilizing a common hardware base It also serves to reduce the inventory of spare parts necessary to achieve a satisfactory level of backup reliability

The following problems are not being addressed at this time except as a secondary consideration

- Complete, extended communication outage within cluster
- Arbitrary and general partitioning within the local

[1] "Distributed Operating System Design Study Final Report" BBN Report No 4671, May 1981

cluster.

- Loss of global (internetwork) communication services

Handling these problems may be important to the command and control environment. However, we believe that their solution is beyond the scope of the current effort.

5 Scalability

The objective in this area is system architecture and design that is cost-effectively scalable over user population sizes ranging from small configurations (e g , tens of users) to large configurations (e g , hundreds of users) The aim is to attain uniform functional and performance characteristics over reasonably scaled versions of the system by adding additional hardware and software capacity without introducing excessive escalation of per user cost and performance or requiring redesign of the system structure

5 1 General Approach

The scalability of a computer system is dependent on many capacity and performance factors ranging from hardware component interconnect structures to high level software resources fabricated through systems programming Due to the off-the-shelf nature of many of the primitive system components being used and the generalized nature of the eventual applications efforts to achieve system scalability must necessarily be focussed on the scalability of the system functions supported by the DOS

In general, system scalability and support for system growth can be somewhat different things Scalability is often

PREVIOUS PAGE IS BLANK

achievable by procuring "larger" units for larger configurations whereas growth is often associated with "additional" units over a period of time Clearly, addressing the growth issues can in many ways, subsume the scalability issues One of the major attractions of a distributed architecture is that it can potentially support growth beyond the limits of conventional systems and hence can attack large scale system scalability from a growth standpoint Additionally we believe it is operationally and logistically more attractive to support scalability needs from an incremental growth viewpoint in order to limit the number of distinct parts and limit the effects of losing a single unit

Our system concept for meeting scalability objectives relies on five main points supporting system growth

1 Adoption of an inexpensive communication architecture which makes it simple to include additional processing elements

2 Selection of modular, inexpensive DOS hardware so that DOS processing elements can be added in small increments as needed without grossly impacting total cost of the system.

3 Careful attention to the potential size estimates for a maximum configuration to ensure that software structures can be made large enough (e g address fields) and that where appropriate, their implementation is partitionable across multiple instances of the function which share the processing and data load

4 Avoidance of so-called N squared solutions which require each element to interact with every other element While these approaches are usually acceptable for smaller configurations they often break down for larger ones

5 Select application systems for inclusion in the demonstration configuration which themselves scale through a range of sizes

5 2 Specific Approach

The selection of a bus communication architecture and Ethernet in particular is in large measure based on providing a simple, underlying basis for system scalability The bus architecture provides a simplified means for supporting a hardware base in which every processing unit can a priori communicate equally well with every other processing unit without regard for routing, processor placement, and other such issues In addition, Ethernet can physically support large numbers of processing units which can be added or removed at will and can also inexpensively support small configurations An important non-goal at this stage of the project is the scalability of the network communication medium itself Any future work in this area will be based on adding an additional Ethernet link to each processing element (also a reliability measure) or on [illegible] network substitution

Low cost incremental expansion also motivates the [illegible] of the M68000-based GCE which will be used as a [illegible] for many I/O functions While it is [illegible] precise number of GCE's required for a minimum [illegible]

approach here is to support some degree of GCE functional multiplexing to be used in small configurations, and to make use of dedicated function units in larger or higher performance configurations. The ability to scale up or down also played a role in selecting application hosts for the initial demonstration environment. Both the UNIX and VMS subsystems are or are expected shortly to be supported on a range of hardware bases both larger and smaller than those for the current configuration.

The VAX which was chosen as one of the major mainframes of the system is a good example of a system which can scale over a wide range. For the initial system configuration will include a VAX 11/750. Without any significant software or peripheral changes, we could substitute any processor from the VAX family with presently includes the VAX 11/780 and VAX 11/782 with an increase in capacity of about two and four respectively. In addition, a VAX 11/730 was recently announced which allows substitution of a smaller and less expensive machine.

The choice of a C70 host represents another kind of provision for scalability. In this case, the desire was to include a computer running an operating system which ports to a variety of machine architectures of varying sizes. The best candidate for operating system portability over medium to small computers is UNIX, so we chose the C70, one of the most

effective computers supporting UNIX We expect that the substitution of another UNIX system would require only a modest effort

Supporting system software scalability implies ensuring adequate or adequately expandable address fields table sizes etc to meet anticipated needs of target configuration sizes It also implies including growth as a factor during the design of the implementation of DOS system functions There are two distinct aspects of a distributed implementation of a given function One aspect is concerned with redundancy as described in the previous section The other is concerned with partitioning responsibility for a function to provide support for a larger client base It is generally easier and hence more desirable to build a self-contained implementation of a function than it is to develop a partitioned implementation since there are fewer error recovery considerations and fewer resource management considerations However to meet our scalability objectives some functions may require a partitioned design for supporting large configurations although they may also be run unpartitioned for small configurations The analysis of the need for a partitioned implementation will be done at design time on a function by function basis We expect that the expansion capabilities of the system will be sufficiently infrequent to allow off-line system reconfiguration time

approaches to many scalability problems

6 Global Resource Management

In many computing environments, and most especially a command and control environment, the administering organization needs some degree of control over the ways in which system resources are allocated to tasks to meet their processing demands. This control is frequently provided by the ability to designate some tasks as more important than other competing tasks, and in the ability to effect automated resource management decisions in an attempt to improve some measure of system performance. These functions are often referred to as "priority service" and "performance tuning" respectively. Most computer systems provide some facilities in these areas and many provide rather elaborate facilities which more than adequately address command and control needs within a single processing node. The objective of this project is to provide support for sustaining these elements of system control in areas that transcend a single processing node.

6.1 Objective

The objective in the area of global resource management is to augment the resource management facilities already present in single node systems with simple, additional mechanisms for supporting various policies of administrative control of

automated distributed resource management decisions The emphasis is on methods for ensuring the prompt completion of important processing tasks and on the distribution of processing load across redundant resources

6.2 General Approach

Global resource management in a communications oriented environment is an area where the system wide ramifications of employing such techniques are yet not completely understood As a consequence, and because of the desire to achieve an operational prototype in a short time frame, we are following a simple, low risk approach The focus of our effort is on those aspects of global system control directly related to the distributed nature of the processing environment In particular, the DOS will focus on the coordination of the priority handling of all parts of any single distributed computation, and on the selection procedures for choosing among replicated, redundant resources present in the DOS cluster DOS global resource management control will be applied only on large grain decisions (e g initiation of a session, opening a file initiating a program) in an effort to simplify the system and limit the communication and processing overhead that would be required for finer-grained global decision making We do not

anticipate the necessity for reevaluating these resource management decisions at finer grains as a potential source of further optimization The system concept is that adequate administrative control will be achievable by controlling the set of tasks which may be competing for resources (load limitation) and by controlling the pattern of use of specific instances of the resource which they will be competing for This is to be accomplished by providing means for administratively limiting the offered load and influencing both the resource selection procedures (where a selection is possible) and the sequencing of the use of the resource after selection using priority The insertion of DOS control points for limiting load, effecting global binding decisions, and controlling order of service are a sufficient set to carry out administrative policy The low risk nature of our effort comes in emphasizing simple mechanisms at these points of control, which in some cases might prove to be suboptimal.

6 3 Specific Approach

The DOS system model is based on active user agents (processes) which access a wide variety of abstract resource types, some of which are directly associated with physical resources (e g , a VAX processor) and others of which may have

distributed implementations built out of composite non-distributed objects All of the resource types have some form of type dependent resource management software associated with them The following three points are important to our global resource management concepts.

1. Every resource request has a "priority" attribute associated with it which is derived from the initiating agent. Although the resource management discipline will be different for different types of objects, the intent of the priority attribute is to provide an object type dependent form of preferential access relative to the use of the resource. Users will have a range of administratively set priorities available for their use To ensure access to the system for potential high priority tasks, system login is a "prioritized" request and may result in preemption of a lower priority user should there be no additional slots. This is accomplished by ensuring that the system "reserves" enough capacity to always accept another login request If the priority of the potential new user exceeds the priority of one of the current users, and if the login would otherwise fail due to lack of available resources a lower priority user will be preempted in favor of the initiation of a new job for a high priority user Once a job is initiated, the current priority of the initiator will determine how the task competes with other active tasks. Other forms of load limitation will be added as necessary as a means of administratively controlling system responsiveness on available resources

2. Automated DOS global resource management decisions will be made predominantly when an agent accesses an object which has multiple instances (e g , multiple processors able to execute the same code, multiple instances of a file, etc) The algorithms for making the selection will be controllable by the "owner" of the composite object Control will be in the form of choosing from a standard set of algorithms supported by the system making use of relevant available data which could include object attributes, collected load data previous selection, first to respond to broadcast, etc

3. We are assuming adequate network transmission capacity when smoothed over reasonably short time frames (i e no continual network overload) This assumption, which seems to be substantiated by early available local network operational experience (albeit not in a command and control environment) makes resource management of the network bandwidth generally unnecessary at this time If scaled load projections indicate potential long term overload situations, our approach for the Ethernet will be to attempt to develop techniques for detecting and limiting the effects of this situation While it is premature to discuss details of such techniques, a promising approach is to attempt to establish a dynamic network transmission priority level, forcing temporary deferral of data transfers below this priority level and providing a means for raising the current level until the overload subsides.

Using these mechanisms, controlling the processing activities of the DOS cluster becomes a policy issue of selecting appropriate priorities and parameters to maximize the ability of the system to meet specific organization objectives

7 Substitutability of System Components

Over the course of time and especially when deployed in non-laboratory operating environments, we anticipate the need to substitute alternative hardware and operating system components which are more appropriate for their environment than those selected for the ADM configuration. It is desirable to be able to alter components in order to match the system characteristics to the needs of operational command and control environments and also to reflect changing availability and cost-effectiveness of components. The ability to perform appropriate substitutions of components in the DOS system is expected to expand the applicability of the DOS system and to lengthen its useful lifetime.

7.1 Objective

The objective in this area is to design the system so as to maximize the potential for component substitution in the system hardware architecture at a later time. System components which are candidates for substitution are the local area network, the GCE configurations, the application hosts, and the gateway

PREVIOUS PAGE IS BLANK

7 2 Approach Use of Abstract Interfaces

The intent of component substitution is to replace a functioning unit with another one capable of performing basically similar operations, but with other properties which make it more attractive or appropriate than the original For example substituting a fiber optic communication network for a coaxial cable network might make sense for a command and control environment concerned with portability or electromagnetic radiation While the basic communication properties of the two systems are equivalent as far as the DOS is concerned environmental considerations might motivate the substitution Similarly, most computer systems can be made to perform a wide range of tasks However, some are judged better than others for certain applications, and hence would motivate the selection of different application hosts to suit the needs of particular command and control applications

Our approach for supporting component substitutability is to define and use appropriate abstractions of the substitutable components as the entity incorporated into the DOS The abstract interfaces are based on common properties of a class of interchangeable components not on specific capabilities of a single component Except under special circumstances unique properties and peculiarities of the hardware will not be visible to the VDM

will be avoided in the definition of abstract interfaces, and where used will be isolated in the code supporting the abstraction to facilitate emulations within other components

Two additional implications fall out of this policy We must expect to lose some efficiency of implementation, since we may need to avoid features that have been built into some components explicitly to solve problems which we may encounter We expect this effect to be small The second side effect of the abstract interface should be increased productivity during the development of the DOS, since an abstract interface is easier to understand and work with This is, in effect the argument used for higher-level programming languages and standards of all kinds The adoption of standards of various kinds as mentioned earlier, also enhances component substitutability by providing abstractions which are already incorporated into many product interfaces

7 3 Approach Specific Interface Plans

This section presents a number of interfaces [illegible] which we plan to employ While this list is not exhaustive we believe it captures the major interfaces on which the issues of substitutability will most depend

The initial version of the DOS is using the Ethernet standard as a communication subsystem We expect to be able to switch between optical fiber and coaxial cable implementations of the Ethernet as may prove desirable based on a cost and availability basis More importantly, our abstract network interface will avoid using features of the Ethernet protocol which are not common to local network technology We expect to use only packet transfer, broadcast and possibly multicast in developing the network abstraction In addition we expect to use IP datagram service as the lowest level IPC abstraction This enhances our independence of the underlying network, and makes it easier to later substitute alternate communication subsystems which can support the abstraction, such as the Flexible Intraconnect

The GCE's represent the implementation base for a number of important DOS functions It is therefore critical that we address the issue of substitutability for the GCE's GCE substitution has two aspects one is the ability to substitute another machine for the present GCE, the second is the ability to substitute for parts of the GCE

We plan to address the first problem the ability to switch GCE's at some future date, by programming in common high level languages to the greatest extent possible We are [illegible]

two languages C and Ada C is a language developed as part of the UNIX system with the goal of being portable to a variety of machines It has largely met that goal although it requires careful attention to coding style to assure the portability of programs written in C <1> . However, there is the possibility of a better choice, Ada, being available in the near future Since Ada is a DOD standard language, its availability on a variety of processors relevant to command and control environments is assured In addition, Ada is a more modern and capable language which should enhance our ability to write code with minimal machine dependency

Substitutability within the GCE is also a matter of concern and attention We are building the GCE strictly out of off-the-shelf components using published and emerging standards to minimize our commitment to any particular part of the GCE For instance, the GCE uses a Multibus bus and backplane which is supplied by a variety of vendors in a wide range of capabilities The processor board is a design developed by Stanford and licensed to at least four manufacturers who are producing compatible boards In addition with only software changes the type of processor board can easily be changed [illegible]

<1> The choice of C was dictated by its immediate availability and the software support already available for the [illegible] processor, a Motorola 68000

probably more different processor boards available for the Multibus than for any other computer bus The use of the Multibus also assures easy substitution of memory Ethernet Controller, I O ports, etc It also assures that any as yet unidentified needs for hardware interfacing can likely be met with off-the-shelf components, due to the popularity of the bus

Our ability to do general substitutions for application hosts is based on our attempts to use portable languages a network (Ethernet) which will soon have interfaces available for a wide range of computer systems, and the concept of a DOS access machine Use of portable languages in the DOS means that we may be able to move software from one DOS host to another The use of an access machine as a means of connecting an application host to the DOS is intended specifically to minimize the effort of host substitution by maximizing the retained software in the access machine GCE Precisely which DOS functions can be handled within the access machine GCE without incurring a similarly complex interaction with the host is yet to be determined

Finally, the most likely substitution to be made in the course of our effort is a substitute for the ARPANET gateway We have adopted the use of an LSI 11 as the gateway to be able to use standard off-the-shelf ARPA internet gateways A [illegible] to the LSI-11 gateway is currently being developed as part of

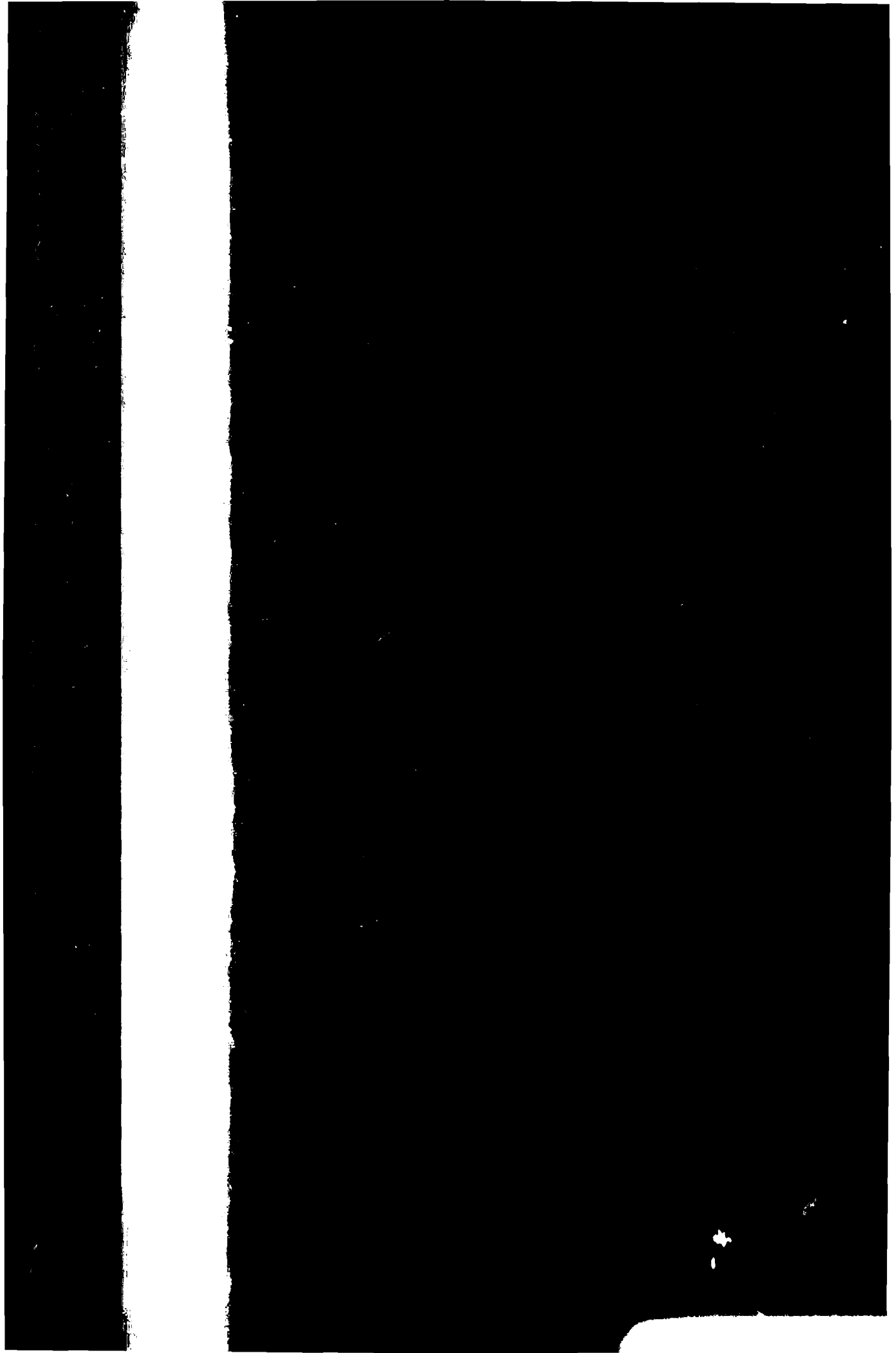
another BBN project. One aspect of our attempt to keep in step with Internet community activities is an anticipated changeover to a new gateway when that development completes. One of the candidate architectures being considered for the future gateway is the equivalent of the DOS GCE

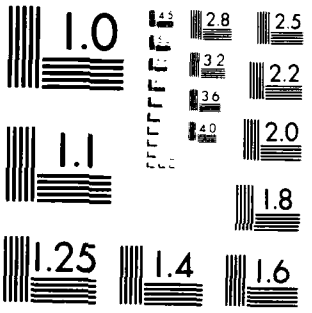
AD-A142 795 CRONUS A DISTRIBUTED OPERATING SYSTEM: FUNCTIONAL DEFINITION AND SYSTEM C..(U) BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA R E SCHANTZ ET AL. FEB 84 BBN-5401 2/2

UNCLASSIFIED RADC-TR-83-254 F30602-81-C-0132 F/G 9/2 NL



END
DATE FILMED
8-84
DTIC





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

8 Operation and Maintenance

It is desirable for the design of any computer system to facilitate the operation and maintenance of the system. In our opinion, this is one of the areas that has not yet received adequate attention, predominatly because few extensively distributed systems have reached operational status. Distributed systems, and especially systems incorporating many heterogeneous parts, are far more complex than their centralized, homogeneous counterparts. Routine chores, such as adding new components to the configuration, coordinating new releases of system software, and initiating diagnostic routines, become much more complex in a distributed system environment. The natural tendency to handle each component separately has shortcomings in the effort required and the sophistication needed to correctly complete simple maintenance activities. The reason for citing operation and maintenance as a goal is our belief that the success of the distributed system concept in Air Force command and control environments will to some extent be dependent on the management of the routine housekeeping chores associated with any computer system.

The objective in this area is to simplify the operation and maintenance procedures for the system so that these tasks are manageable by personnel other than system programmers.

PREVIOUS PAGE IS BLANK

9 Test and Evaluation

One of the important aspects of introducing new system concepts or approaches is the need to answer the question of how successful they have been in meeting their objectives. The test and evaluation phase of our project is intended to provide these answers. We include a discussion of test and evaluation in this "early" project documentation to emphasize our approach of applying considerations in this area throughout the project. Test and evaluation should be more than an after-the-fact activity and can be a positive factor in driving the design and the implementation.

We can identify four distinct stages, spanning the project lifetime, that are relevant to test and evaluation.

1. Setting goals. Section 3 outlined the approach and named the three primary goals for which prior test and evaluation methodologies will be developed, namely, coherence and uniformity, survivability and integrity, and scalability.

2. Defining test and evaluation methodologies. In parallel with the system design, test and evaluation procedures will be developed for the three primary goals. Insofar as practical, these procedures will each define a "figure of merit" for their respective aspects of the design and implementation, and an effective means for determining the figure of merit. In some cases, the need to carry out these tests may influence the system implementation to more effectively support evaluation.

3. Extended system test. During the last few months of the contract period of performance, the system will be subjected to an extended test phase. Operational testing will be done by monitoring the DOS ADM as it is used by the

system developers and other groups which may be solicited to build example application systems. synthetic testing will be done through the use of synthetic workload generators for reliability and scalability testing

4. Reporting. The results of the extended system test will be analyzed and judged by means of the yardsticks defined in the second stage. Documentation will be prepared which reflects the results of the test and evaluation phase.

The following sections discuss our current view of the test and evaluation issue as it relates to each of the primary DOS goals.

9.1 Coherence and Uniformity

A system is coherent if the system concepts "play together". coherence makes a system easier to understand and use. A system is uniform if different components perform the same or similar functions in the same or similar ways. Both coherence and uniformity are largely subjective measures of a system, and thus our test and evaluation procedures for this goal will be to gather and analyze the subjective reactions of the user population at the end of the extended test period. Users will be asked to evaluate the system both on absolute terms (what they liked and didn't like) and on a relative basis (comparing the file system, for example, to the UNIX file system) Users will be asked to respond to questions in specific areas, and will also be given an opportunity for open-ended comment The user

statements will be collected, digested, annotated and presented in an organized format <1> .

We anticipate that the user population available for system evaluation will consist of two, probably overlapping, groups the system developers, and one or more groups selected to develop exemplary application and demonstration programs. There is, of course, a special motivation in requiring the system developers to use the system in the normal course of their work--the feedback path from user to developer is minimized. Design decisions which cause great difficulties will be rapidly exposed and revised. The system developers are also likely to be more tolerant than other users of small "rough edges", which means that they can begin to use the system earlier, before the polishing is finished. This practice generally encourages the developers to be prompt, careful, and down-to-earth, because their own productivity is at stake. A consequence of this is that the initial services developed for the system will be oriented toward the needs of the system developers. In many cases (e.g., text editing) these services have utility in other environments. In those cases where utility is limited to system

---

<1>. The paper "Reflections in a pool of processors--an experience report on C.mmp/Hydra", W. A. Wulf and S. P. Harbison, AFIPS Proceeding of the National Computer Conference V47, 1978, is an interesting example of an evaluation of this type, and will serve as a model.

developers, they do form the foundation of supporting the enhancement of the DOS system through it own facilities.

The system developers will further test the system design through the implementation of some system services, such as file archiving and command language interpreters, as application level programs. The implementation of these services will test the ability of the DOS to support such system functions without resorting to modifications of the software within the DOS security envelope. Minimizing the amount of software within the security envelope is a problem analogous to minimizing the size of a security kernel in a conventional, single-host operating system; thus experience gained relating to this aspect of system extensibility is especially important.

The experiences of the system developers, however, are no substitute for those of application programmers. Application programmers can be expected to make demands upon the completeness and accuracy of the documentation, for example, and to exercise the system in ways that were not anticipated, or not often used by the developers. Because application programmers will lack in-depth knowledge of the DOS implementation strategies, their reactions are an important test of the user-level conceptual models defined in the user manuals. Due to limited time and effort, only small-scale examples will be constructed during the

extended system test, but these can nonetheless be expected to yield significant insight into the usefulness of the DOS design and implementation.

9.2 Integrity and Survivability

The test and evaluation of integrity and survivability of a system is one of the harder to perform. First, one must decide what constitutes appropriate behavior in this area, and then one must design (non-destructive) methods of test.

The first step in the test and evaluation procedure for system integrity and survivability is to ensure that the failure modes identified in Section 4 can be artificially and easily induced in the ADM. For the failure of a processor, for example, this may mean simply that the processor can be either physically or logically disconnected from the network.

The monitoring capabilities of the DOS will include the maintenance of online error logs. These log files will be utilized during the extended test phase to record naturally occurring failures within the ADM, as the DOS is used routinely by the development team and application programmers. Errors which cannot be automatically recorded because of the nature or

extent of the failure will be manually recorded in an offline log.

Finally we intend to build one or more reliable applications, and exercise the applications by means of aritificially induced failures.

9.3 System Scalability

There are two important facets to the evaluation of DOS scalability: function and performance. By scaling of function we mean the ability of the various DOS mechanisms to scale to larger configurations and user populations without regard to the effect of scaling on performance. Typically, different mechanisms have different limits to scaling, which are determined by a sequence of decisions during design and implementation. In a conventional single-host operating system, these limits are often real constraints on the range of applicability of the system. For example, an operating system might limit the number of active users or the maximum file size. The first, and easiest, part of the evaluation of scalability is the identification and analysis of these maximum limits to growth.

Even if it is functionally possible to scale the system

along some dimension, such as the number of active users, it may be undesirable to do so on performance grounds. A thorough evaluation of the effects of scaling on performance is not possible within the period of this contract, nonetheless, we expect to obtain some preliminary results by means of direct measurements and performance modeling.

We are interested in two primary dimensions of scaling:

1. Workload scaling. Given a fixed DOS configuration and a well-defined workload, how do the system response times for different classes of users change as the user population increases?

2. Configuration scaling. Given a well-defined workload and a fixed-size user population, how do the system response times for different classes of users change as the number of service hosts is scaled?

One important constraint on the evaluation of scalability is the size of the Advanced Development Model configuration. Because we expect functional limits to the number of hosts, for example, to be on the order of 1,000 <1>, but will have only about 10 hosts (including DOS service hosts) in the ADM, empirical tests of configuration scaling will be possible only over a small portion of the DOS configuration space.

Our approach to the evaluation of scalability with respect to performance will be based on empirical performance data

<1> The Ethernet specification limits the number of attached hosts to approximately 1,000.

obtained from the ADM, used as the basis for system models which extrapolate to much larger workloads and configurations. By its nature, this type of performance modeling cannot be extremely precise, and tends to be more useful as a qualitative indicator of feasibility rather than a quantitative predictor of system performance. Analytic models can be constructed and evaluated very rapidly, so they are an inexpensive tool to apply. We believe they are the most appropriate modeling technique during the early life of a system, when decisions are more apt to concern gross changes in resource management strategies than fine-tuning of algorithm parameters.

The DOS system monitoring facilities will be designed to accumulate the performance data necessary for modeling during routine operation of a DOS cluster. This performance data can be collected during actual use of the system, or while system and application functions are exercised by artificially induced workloads. At this time, it is not known whether data from naturally-occurring workloads will suffice, or whether synthetic workload generators will be required. this issue should be clarified by the definition of the scalability test criteria during the design and implementation phases of the project

ort

as a baseline for command and
to the development of a DOS It
system/network model for a TAFIIS
bal long-haul network) and MINI-
S we are developing in this
e MINI-DOS issues.

I Model

a set of services which should be

f data.

sers.

ons is typical of the application

H. Ruspini, and Christine A
ted Operating System Study Final
-045, Operating Systems, Inc.

ted Information System

programs and processing load whic

The OSI report describes sev
provide starting points for much
design. In particular, they iden
mechanisms as pertinent to the de

- Directory services.
- Allocation of resources sha
- Scheduling of tasks involvi
- Access to global system sof
- Performance monitoring.
- Degradation handling and sy
- Interprocessor communicatio
- Multi-level data security.

With the exception of multi-level
the scope of this project, our sy
areas. The first two items, dire
resources, are at the heart of ou
critical to the design of a local
is to operate as an integrated un

Our major extension to the
on the MINI-DOS aspects of the ar
gateway functions to link instanc
preliminary step to addressing MA

communication speed, delay, reliability and security in the MAXI-DOS area change the nature of the network integration task, making it distinct from MINI-DOS system integration

10.2 OSI Identified Functions

The OSI report identifies a number of important functions of the DOS. In this section we briefly indicate our approach as to these functions, and contrast them with potential approaches suggested by the OSI report.

Interprocessor communication will be provided in the DOS using Ethernet together with DOD standard interprocess communication protocols. The Ethernet includes cable network hardware together with a local net CSMA/CD protocol. Above the Ethernet layer we will be using Internet Protocol (IP) Datagrams and, where reliable connection-based transport is required, Transmission Control Protocol (TCP). The use of IP and TCP within the cluster assures a degree of IPC compatibility with the Internet community and with other DOD systems. We selected a high-bandwidth local network, since we believe high bandwidth, low delay transmission is necessary in order to enable the DOS to operate in an integrated fashion. The OSI report does not focus on MINI-DOS interprocessor communication. It suggests only that the MAXIDOS be capable of 10-50 Kb second, similar to our ARPANET

gateway but two
local network

Resource M
the OSI report.
be tightly cont
a high bandwidt
suggested in th
centralized W
communication m
abstraction MIN
distributed if

Security a
since the OSI r
local net (cell
calls for a gen
mechanism, whic
schemes. Howeve
security

Configu
the OSI report
environment, wh
capability and
recognizing thi

gateway but two orders of magnitude less than the speed of our local network

Resource Management in the MINI-DOS is left unspecified in the OSI report, except for indications that resource management be tightly controlled and many appropriate strategies may require a high bandwidth communication medium. In addition, it is suggested in the report that resource management will probably be centralized. We are, of course, providing a high-bandwidth communication medium in the Ethernet. At higher levels of abstraction MINI-DOS resource management implementations must be distributed if the system is to survive component outages.

Security approaches within the MINI-DOS were not specified, since the OSI report felt it was dependent on the nature of the local net (cell, in the OSI terminology). Our system concept calls for a general purpose access control and authentication mechanism, which borrows from several traditional access control schemes. However, we are not planning to implement multi-level security.

Configuration management was regarded by the authors of the OSI report as a problem largely restricted to the MAXI-NET environment, where noisy channels might eliminate communication capability and isolate local networks from each other. While recognizing this problem, we believe that configuration

management is also an important issue within the MINI-DOS, where individual host failures should not be allowed to disrupt the local network. In our system concept, there are two levels of configuration issues: the reconfiguration requirements resulting from failed components (and of components brought back into service), and reconfiguration resulting from scaling of the system, planned growth, and phasing in and out of generations of equipment.

The first type of reconfiguration, resulting from system faults, can to a great extent be handled by automatic procedures. These procedures require mechanisms which operate correctly *despite outages of components, and of mechanisms* which perform automatic reconfiguration when failures are recognized.

The second type of reconfiguration will be provided by manual intervention. Manual updates to configuration tables will be sufficient to accommodate many anticipated changes, and careful, modular system design should enable us to keep more radical configuration changes localized within modules.

Data Base Management is recognized as an important function within the DOS cluster, but it is largely separable from the design of the DOS itself, and therefore is outside the scope of the present effort. The DOS will provide basic support for data storage and access, including reliable file mechanisms.

which could provide a reasonable base for the implementation of data base management systems. In addition, an alternate approach to data base functions, dedicated data base machines, is now emerging. This approach fits in well with our DOS system concept of dedicated function components and we recommend that an instance of such a system be considered for inclusion in the DOS configuration. One of the issues we see concerns the potential conflict of the "black box" nature of these machines, and the desire for integration with other DOS system concepts (e.g. resource management, reliability).

11 DOS Glossary

Abstract Object Model
Model of entities manipulated by the DOS which attempts to treat a wide variety of differing system and user entities in a unified manner. Types of DOS objects will include files, devices, and processes. Associated with each object is a unique identifier, and services for cataloging and controlling access.

Access Point
Point of interface between the user and the DOS. The access point for a DOS user may be a Terminal Access Controller (TAC), a workstation, or a DOS application host.

Address
Bit string representing the location at which an object may be referenced. Addresses often consist of several concatenated fields, representing a hierarchy of containing "locations". Rome is in New York in the United States of America. A field designates a unique location in the locale containing it; fields may be reused in different locales. Rome is in Italy.

Advanced Development Model (ADM)
Physical instance of the DOS to be developed under the DOS Design/Implementation contract. the ADM will initially be used by the system developers.

Application Host
DOS host on which application programs run. There are potentially many types of application hosts in the DOS. In the ADM, two important types are general-purpose timesharing hosts and GCE's dedicated to application programs.

Capability
If a process possesses a capability for an operation on an object, it may invoke the operation against the object. Possession of the capability is proof of authorization--no further access control check is made.

Cluster
The local network and its hosts. The cluster is the main focus of DOS integration activity. A primary characteristic of a cluster is its uniform high-speed

low delay communication

Essential Service
: Service of the DOS required for the continued operation of the DOS. Essential services are candidates for continuous availability, which is provided through redundancy.

Generic Computing Element (GCE)
: A small computer system made up of interchangeable parts upon which many DOS functions will be built. In the Advanced Development Model of the DOS, GCE's will be built using 68000 processors in a Multibus backplane.

Integrity
: Maintenance of system and application state information in a consistent state, meeting the system and application program functional specifications Emphasis is on the maintenance of system integrity across failures, i.e., the phases of failure detection, isolation, and recovery.

Process
: Model of the active agent or instruction execution in the DOS. Processes in the DOS are objects, and will provide a DOS-wide mechanism for addressing, invocation, and control.

Primitive Process
: Simple version of process which provides only a limited set of control functions. It is presumed that any host in the DOS will be able to provide a base for the implementation of at least one primitive process

Scalability
: The capability of the DOS to grow or shrink in size within reasonable bounds Scalability will be supported by two means, the replacement of processors with more (less) capacity and the addition (deletion) of processors.

Security Envelope
: Boundary around the DOS cluster delimiting the region of the system within which security is ensured by the use of unforgeable addresses and trusted agents Outside the security envelope, capabilities and passwords will be used to authenticate DOS access

Survivability
: Ability of a system to continue to perform a given function despite expected failures, with only insignificant performance or functional degradation, synonymous with "continuous availability".

Symbolic Name
: Identification of a DOS object in a global name space independent of the object's location or the location of the reference. The symbolic name space is designed to consist of character strings, and is easily manipulated by the users of the system. A mapping is provided through the catalog mechanism for translating symbolic names to universal identifiers.

Universal Identifier (UID)
: A fixed-length bit-string which identifies, or names, a unique object. Every DOS object has a universal identifier, no two objects have the same identifier.

Workstation
: A computer which is dedicated to single-user-at-a-time operation, which provides both computational services and an access point to the DOS. In the Advanced Development Model, Jerichos will fulfill this role.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.


DATE
ILME

recognizing thi

